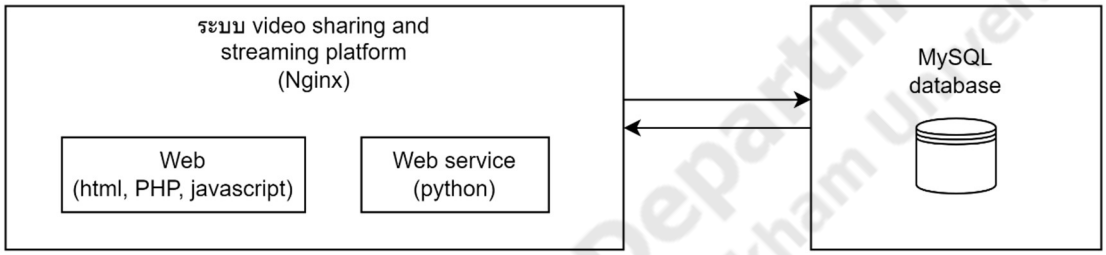
บทที่ 3

ขั้นตอนการดำเนินงาน

3.1 กรอบการดำเนินงาน

กรอบการทำงานนี้จะแสดงขั้นตอนการพัฒนาระบบ เว็บแอปพลิเคชันแบ่งปันและสตรีมวิดีโอ ซึ่งมีการดำเนินงานดังนี้



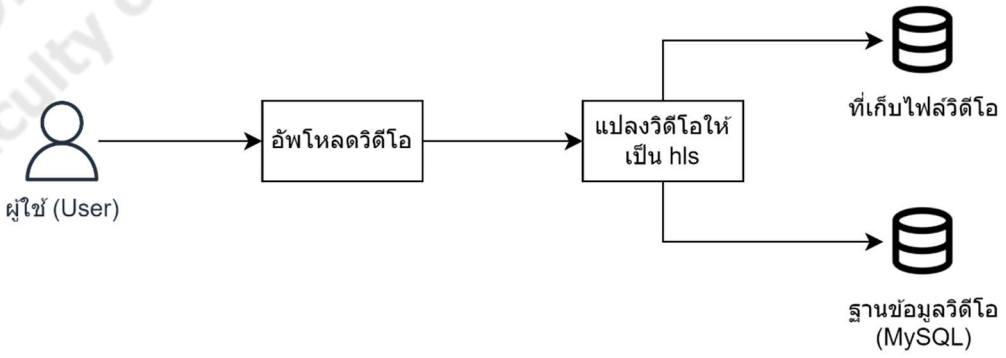
ภาพประกอบที่ 3.1 กรอบการดำเนินงาน

3.1.1 การเก็บข้อมูลทั่วไป

ข้อมูลทั่วไปเช่น ข้อมูลผู้ใช้ ข้อมูลวิดีโอทั่วไป ข้อมูลประวัติการรับชม จะถูกเก็บไว้ใน MySQL database

3.1.2 การเก็บข้อมูลไฟล์วิดีโอ

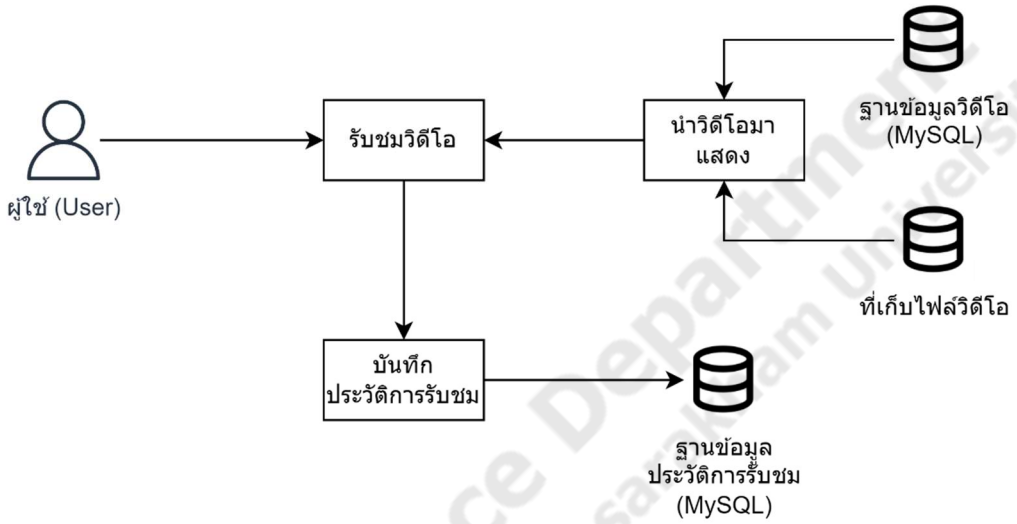
ไฟล์วิดีโอหลังจากการอัปโหลดไฟล์ไปยังเซิร์ฟเวอร์จะถูกแปลงไฟล์เป็น hls และจะทำการเก็บบน directory ของเซิร์ฟเวอร์



ภาพประกอบที่ 3.2 แสดงกระบวนการเก็บข้อมูลวิดีโอ

3.1.3 การดูวิดีโอ

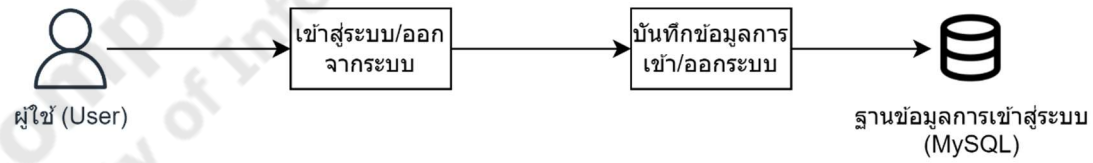
เมื่อผู้ใช้ต้องการดูวิดีโอระบบจะนำข้อมูลทั่วไปของวิดีโอจากฐานข้อมูล MySQL และไฟล์วิดีโอ hls จากเครื่องเซิร์ฟเวอร์มาแสดงให้ผู้ใช้รับชม เมื่อผู้ใช้รับชมวิดีโอระบบจะทำการบันทึกประวัติการรับชมลงในฐานข้อมูล



ภาพประกอบที่ 3.3 แสดงกระบวนการดูวิดีโอและบันทึกประวัติการรับชม

3.1.4 การเก็บข้อมูลการเข้า/ออกระบบของผู้ใช้

ทุกครั้งที่ผู้ใช้เข้าหรือออกจากระบบ ระบบจะทำการบันทึกการกระทำและเวลาที่เกิดขึ้นลงในฐานข้อมูล MySQL



ภาพประกอบที่ 3.4 แสดงกระบวนการบันทึกข้อมูลการเข้า/ออกระบบของผู้ใช้

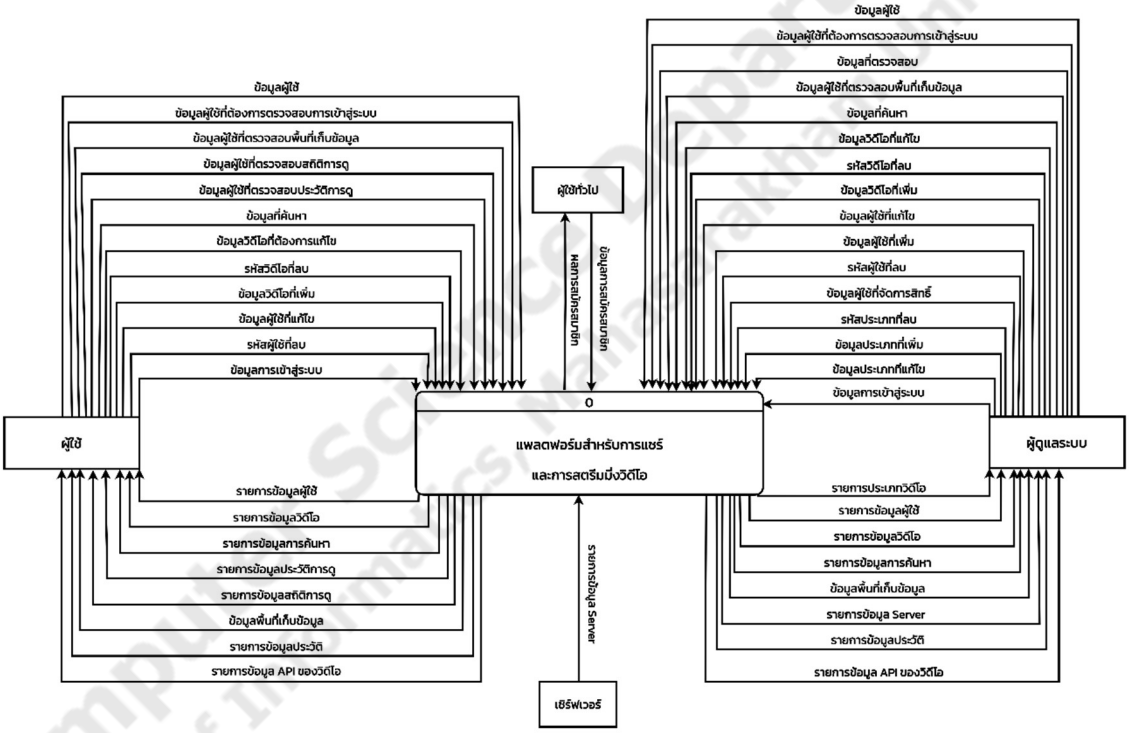
3.1.5 การดึงข้อมูลทรัพยากรของเครื่องเซิร์ฟเวอร์

ผู้ดูแลระบบสามารถดูข้อมูลทรัพยากรของเครื่องเซิร์ฟเวอร์ได้จาก Python flask API

ภาพประกอบที่ 3.5 แสดงกระบวนการดูข้อมูลทรัพยากรของเครื่องเซิร์ฟเวอร์

3.2 การออกแบบระบบ

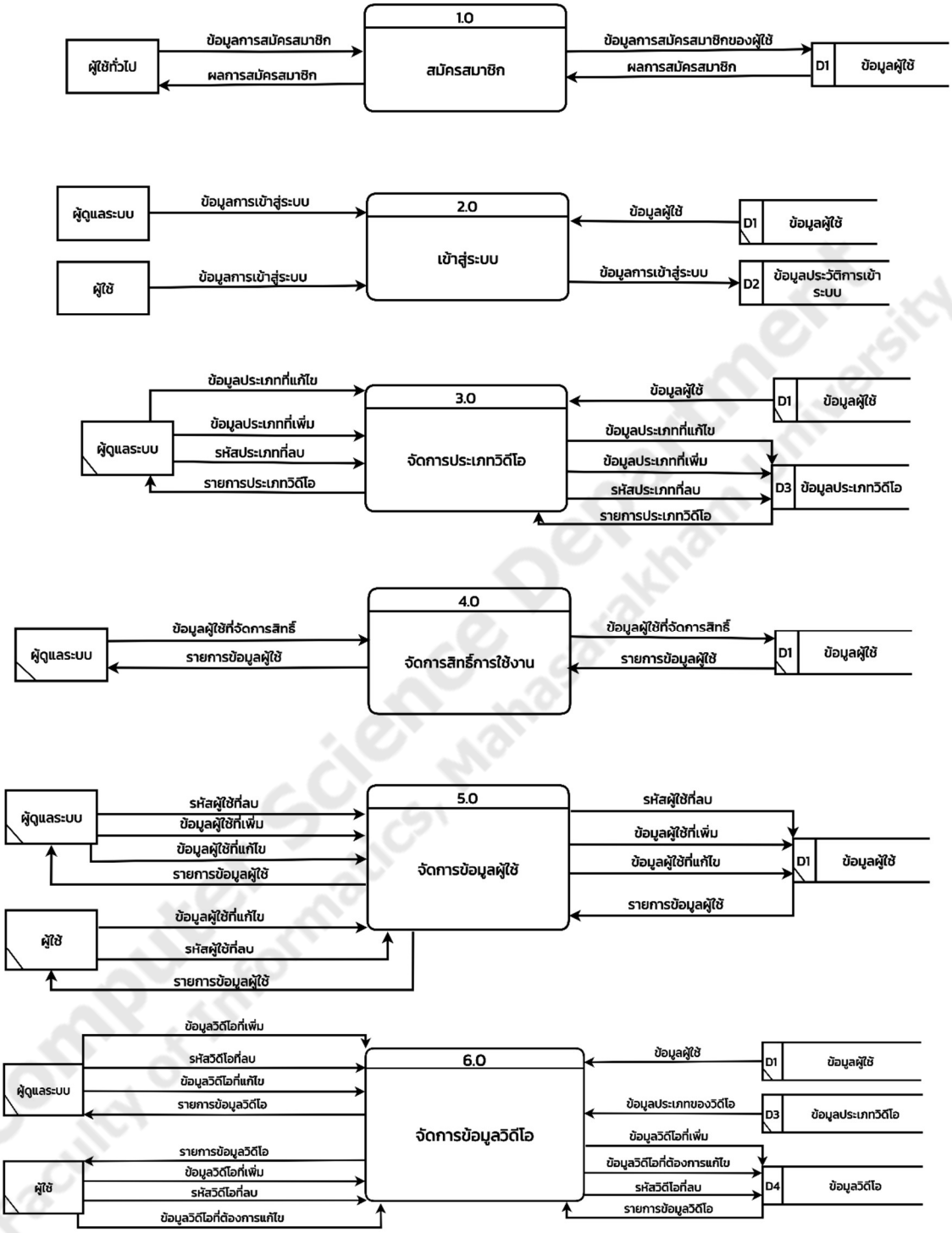
3.2.1 แผนภาพการไหลของข้อมูล (Context Diagram)



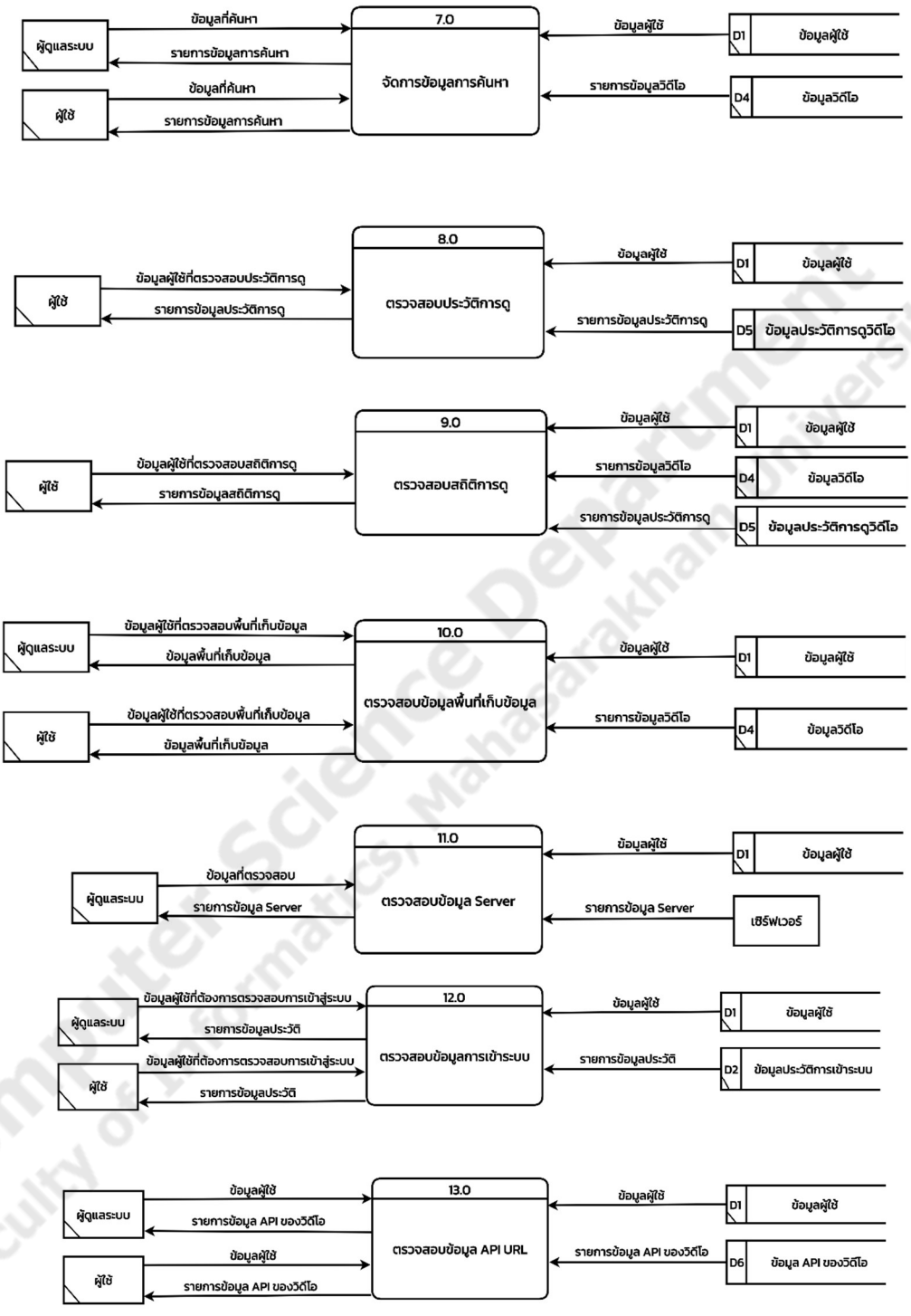
ภาพประกอบที่ 3.6 แผนภาพ context diagram

3.2.2 Data Flow Diagram Level 1

แผนภาพกระแสข้อมูลในระดับที่แสดงขั้นตอนการทำงานหลักทั้งหมด (Process หลัก) ของระบบแสดงทิศทางการไหลของ Data Flow และแสดงรายละเอียดของแหล่งจัดเก็บข้อมูล (Data Store)



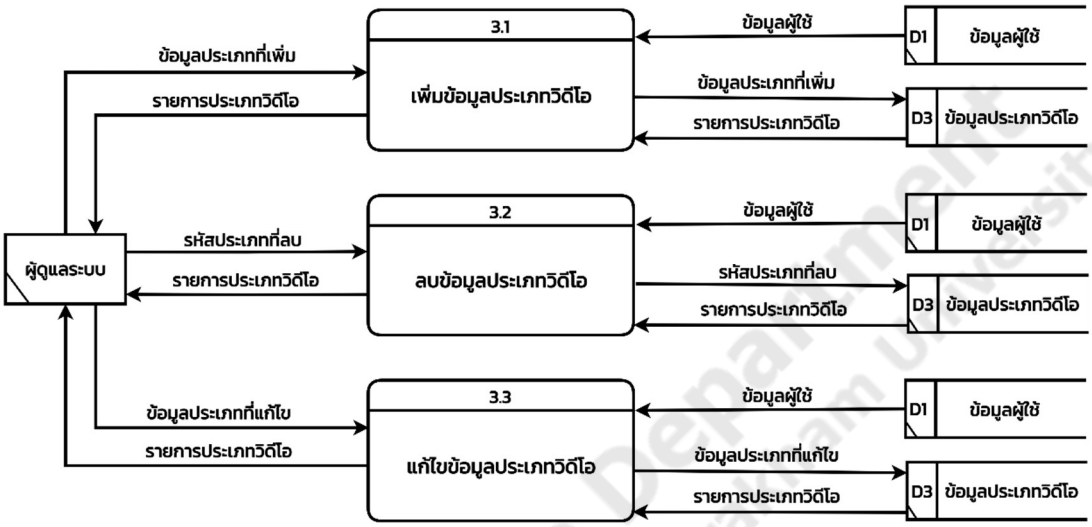
ภาพประกอบที่ 3.7 Data Flow Diagram Level 1



ภาพประกอบที่ 3.7 Data Flow Diagram Level 1 (ต่อ)

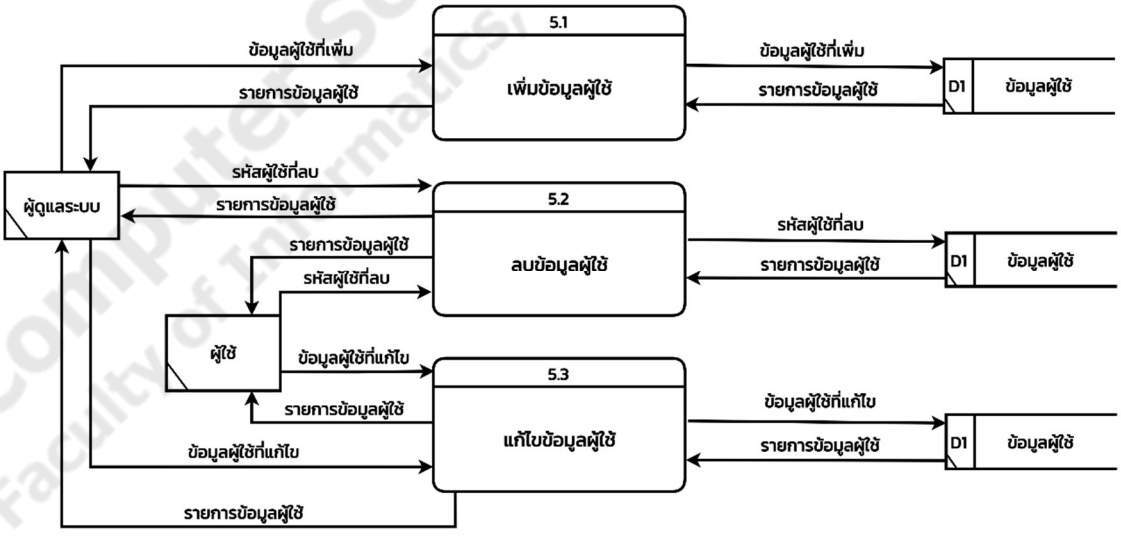
3.2.3 Data Flow Diagram Level 2

3.2.3.1 แผนภาพการไหลของข้อมูลระดับ 2 กระบวนการที่ 3



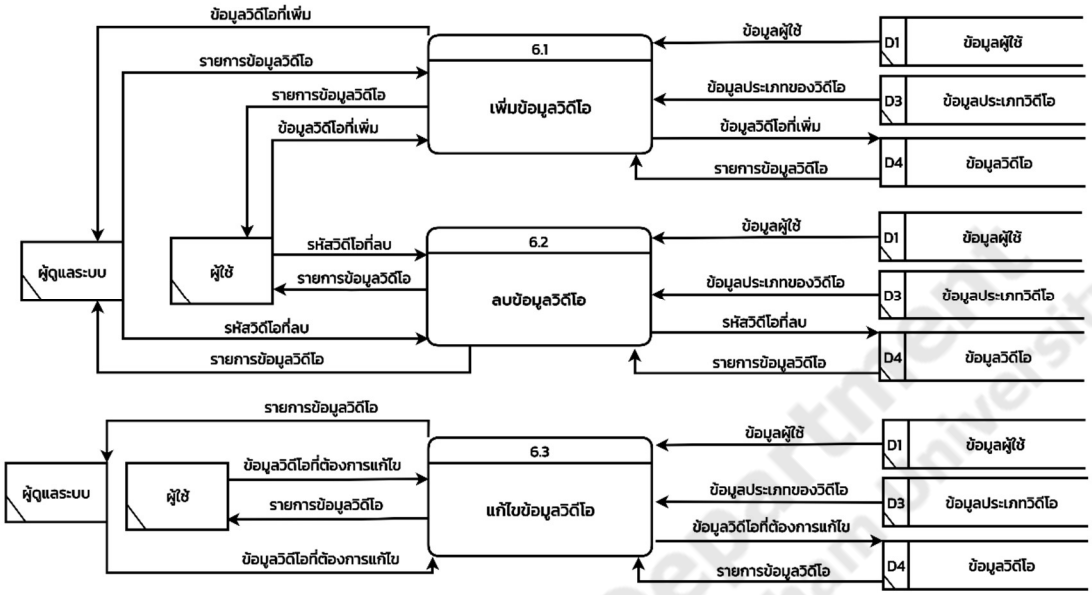
ภาพประกอบที่ 3.8 Data Flow Diagram Level 2 Process 3.

3.2.3.2 แผนภาพการไหลของข้อมูลระดับ 2 กระบวนการที่ 5



ภาพประกอบที่ 3.9 Data Flow Diagram Level 2 Process 5

3.2.3.3 แผนภาพการไหลของข้อมูลระดับ 2 กระบวนการที่ 6



ภาพประกอบที่ 3.10 Data Flow Diagram Level 2 Process 6

3.3 Data Store

ตารางที่ 3.1 Data Store

ID	Data Store	Description	Data Structure
D1	ข้อมูลผู้ใช้	เก็บข้อมูลรายละเอียดของผู้ใช้และผู้ดูแลของระบบ	รหัสผู้ใช้ + ชื่อผู้ใช้ + อีเมลผู้ใช้ + รหัสผ่านผู้ใช้ + ประเภทผู้ใช้ + จำนวนของวิดีโอผู้ใช้ + รูปโปรไฟล์ผู้ใช้ + เวลาที่สร้างบัญชี + สิทธิ์ในการอัปโหลดวิดีโอ + ชื่อแฟ้มเก็บข้อมูลผู้ใช้ + รูปหน้าปกผู้ใช้ + พื้นที่เก็บข้อมูลผู้ใช้
D2	ข้อมูลประวัติการเข้าระบบ	เก็บรายละเอียดของการเข้าสู่ระบบและออกจากระบบของผู้ใช้	รหัสข้อมูล Log + รหัสผู้ใช้ + ประเภทของ Log + เวลาของการเก็บ Log
D3	ข้อมูลประเภทวิดีโอ	เก็บรายละเอียดประเภทของวิดีโอ	รหัสประเภทวิดีโอ + ชื่อประเภทวิดีโอ

ตารางที่ 3.1 Data Store (ต่อ)

ID	Data Store	Description	Data Structure
D4	ข้อมูลวิดีโอ	เก็บรายละเอียดต่างๆ ของวิดีโอ	รหัสวิดีโอ + ชื่อวิดีโอ + จำนวนการดู + ความยาววิดีโอ + ขนาดของวิดีโอ + เวลาที่ Upload + ภาพตัวอย่าง + รหัสผู้ใช้เจ้าของวิดีโอ + สิทธิ์ของวิดีโอ + ชื่อไฟล์วิดีโอที่ผ่านการแปลง + ขนาดความชัดของวิดีโอ + คำอธิบายของวิดีโอ
D5	ข้อมูลประวัติการดูวิดีโอ	เก็บรายละเอียดประวัติการดูวิดีโอของผู้ใช้	รหัสประวัติการดู + รหัสผู้ใช้ + รหัสวิดีโอ + เวลาที่ดูวิดีโอ + เวลาที่ดูค้างไว้
D6	ข้อมูล API URL	เก็บรายละเอียด API URL วิดีโอของผู้ใช้	รหัสข้อมูล API + URL Token + เวลาที่สร้าง + เวลาหมดอายุ + รหัสผู้ใช้ + รหัสวิดีโอ

3.4 External Entity Description

ตารางที่ 3.2 External Entity Description

Name	Description	Input Data Flow	Output Data Flow
ผู้ใช้ทั่วไป	ผู้ใช้ทั่วไปที่ยังไม่เป็นสมาชิก โดยสามารถใช้งานเว็บไซต์ได้บางส่วน รวมถึงสามารถสมัครสมาชิกเพื่อใช้งานระบบได้	- ผลการสมัครสมาชิก	- ข้อมูลการสมัครสมาชิก
ผู้ใช้	ผู้ใช้งานที่เป็นสมาชิกสามารถลงชื่อเข้าใช้	- รายการข้อมูลผู้ใช้ - รายการข้อมูลวิดีโอ - รายการข้อมูลการค้นหา	- ข้อมูลการเข้าสู่ระบบ - ข้อมูลผู้ใช้ที่แก้ไข - รหัสผู้ใช้ที่ลบ

ตารางที่ 3.2 External Entity Description (ต่อ)

Name	Description	Input Data Flow	Output Data Flow
	และเข้าใช้ระบบของเว็บไซต์ได้ทุกระบบ	- รายการข้อมูลประวัติการดู - รายการข้อมูลสถิติการดู - ข้อมูลพื้นที่เก็บข้อมูล - รายการข้อมูลประวัติ - รายการข้อมูล API ของวิดีโอ	- ข้อมูลวิดีโอที่เพิ่ม - รหัสวิดีโอที่ลบ - ข้อมูลวิดีโอที่ต้องการแก้ไข - ข้อมูลที่ค้นหา - ข้อมูลผู้ใช้ที่ตรวจสอบประวัติการดู - ข้อมูลผู้ใช้ที่ตรวจสอบสถิติการดู - ข้อมูลผู้ใช้ที่ตรวจสอบพื้นที่เก็บข้อมูล - ข้อมูลผู้ใช้ที่ต้องการตรวจสอบการเข้าสู่ระบบ - ข้อมูลผู้ใช้
ผู้ดูแลระบบ	ผู้ดูแลระบบ สามารถเพิ่มข้อมูล แก้ไขข้อมูล และตรวจสอบสถานะของข้อมูลต่างๆ ของผู้ใช้งานบนเว็บไซต์ได้	- รายการประเภทวิดีโอ - รายการข้อมูลผู้ใช้ - รายการข้อมูลวิดีโอ - รายการข้อมูลการค้นหา - ข้อมูลพื้นที่เก็บข้อมูล - รายการข้อมูล Server - รายการข้อมูลประวัติ - รายการข้อมูล API ของวิดีโอ	- ข้อมูลการเข้าสู่ระบบ - ข้อมูลประเภทที่แก้ไข - ข้อมูลประเภทที่เพิ่ม - รหัสประเภทที่ลบ - ข้อมูลผู้ใช้ที่จัดการสิทธิ์ - รหัสผู้ใช้ที่ลบ - ข้อมูลผู้ใช้ที่เพิ่ม - ข้อมูลผู้ใช้ที่แก้ไข - ข้อมูลวิดีโอที่เพิ่ม - รหัสวิดีโอที่ลบ - ข้อมูลวิดีโอที่แก้ไข - ข้อมูลที่ค้นหา - ข้อมูลผู้ใช้ที่ตรวจสอบพื้นที่เก็บข้อมูล - ข้อมูลที่ตรวจสอบ

ตารางที่ 3.2 External Entity Description (ต่อ)

Name	Description	Input Data Flow	Output Data Flow
			- ข้อมูลผู้ใช้ที่ต้องการตรวจสอบการเข้าสู่ระบบ - ข้อมูลผู้ใช้
เซิร์ฟเวอร์	เซิร์ฟเวอร์ที่ใช้ในการให้บริการเว็บแอปพลิเคชันและสามารถร้องขอเพื่อดูข้อมูลต่างๆ เกี่ยวกับ Server ได้		- รายการข้อมูล Server

3.5 Data Flow Description and Data Structure of Data Flow

เป็นขั้นตอนการทำงานของเว็บไซต์ซึ่งทำให้เราทราบถึงการรับ-ส่งข้อมูล แสดงถึงการไหลของข้อมูลทั้งข้อมูลเข้า (Input) และข้อมูลส่งออก (Output) ระหว่างข้อมูลต้นทางถึงข้อมูลปลายทาง โดยสามารถอธิบายข้อมูลและขั้นตอนในการดำเนินงาน ดังต่อไปนี้

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow

Name	Description	Source	Destination	Data Structure
ข้อมูลการสมัครสมาชิก	รายละเอียดที่ใช้ในการสมัครสมาชิกของผู้ใช้	ผู้ใช้ทั่วไป	Process 1.0 สมัครสมาชิก	ชื่อผู้ใช้ + อีเมล + รหัสผ่าน
		Process 1.0 สมัครสมาชิก	D1 ข้อมูลผู้ใช้	รหัสผู้ใช้ + ชื่อผู้ใช้ + อีเมล + รหัสผ่าน + ประเภทผู้ใช้ + จำนวนวิดีโอ + ข้อมูลรูปภาพโปรไฟล์ + เวลาที่สร้างบัญชี + สิทธิ์ในการอัปโหลด + แฟ้มข้อมูลในการอัปโหลด +

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
				ข้อมูลภาพหน้าปก +พื้นที่หน่วยความจำที่ใช้
ผลการสมัครสมาชิก	ผลการสมัครบัญชีของผู้ใช้	D1 ข้อมูลผู้ใช้	Process 1.0 สมัครสมาชิก	ผลการสมัครผู้ใช้
		Process 1.0 สมัครสมาชิก	ผู้ใช้ทั่วไป	
ข้อมูลผู้ใช้	ข้อมูลของสมาชิกที่ใช้ในระบบ	D1 ข้อมูลผู้ใช้	Process 2.0 สมัครสมาชิก	รหัสผู้ใช้ + ชื่อผู้ใช้ + อีเมล + รหัสผ่าน + ประเภทผู้ใช้ + จำนวนวิดีโอ + ข้อมูลรูปภาพโปรไฟล์ + เวลาที่สร้างบัญชี + สิทธิ์ในการอัปโหลด + แฟ้มข้อมูลในการอัปโหลด + ข้อมูลภาพหน้าปก +พื้นที่หน่วยความจำที่ใช้
		D1 ข้อมูลผู้ใช้	Process 3.0 จัดการประเภทวิดีโอ	
		D1 ข้อมูลผู้ใช้	Process 6.0 จัดการข้อมูลวิดีโอ	
		D1 ข้อมูลผู้ใช้	Process 7.0 จัดการข้อมูลการค้นหา	
		D1 ข้อมูลผู้ใช้	Process 8.0 ตรวจสอบประวัติการดู	
		D1 ข้อมูลผู้ใช้	Process 9.0 ตรวจสอบสถิติการดู	
		D1 ข้อมูลผู้ใช้	Process 10.0 ตรวจสอบข้อมูล	

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
			พื้นที่เก็บข้อมูล	
		D1 ข้อมูลผู้ใช้	Process 11.0 ตรวจสอบข้อมูล Server	
		D1 ข้อมูลผู้ใช้	Process 12.0 ตรวจสอบข้อมูลการเข้าระบบ	
		D1 ข้อมูลผู้ใช้	Process 13.0 ตรวจสอบข้อมูล API URL	
		ผู้ใช้	Process 13.0 ตรวจสอบข้อมูล API URL	
		ผู้ดูแลระบบ	Process 13.0 ตรวจสอบข้อมูล API URL	
ข้อมูลการเข้าสู่ระบบ	ข้อมูลที่ผู้ใช้ใช้ในการเข้าสู่ระบบ	ผู้ดูแลระบบ	Process 2.0 สมัครสมาชิก	อีเมล + รหัสผ่าน
		ผู้ใช้	Process 2.0 สมัครสมาชิก	
		Process 2.0 สมัครสมาชิก	D2 ข้อมูลประวัติการเข้าระบบ	รหัสข้อมูล Log + รหัสผู้ใช้ + สถานะการเข้าระบบ + เวลาในการเข้าระบบ
รายการประเภทวิดีโอ	รายการข้อมูลของประเภทวิดีโอที่มีในระบบ	D3 ข้อมูลประเภทวิดีโอ	Process 3.0 จัดการประเภทวิดีโอ	{ รหัสประเภทวิดีโอ + ชื่อประเภทวิดีโอ }
		Process 3.0	ผู้ดูแลระบบ	

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
		จัดการประเภทวิดีโอ		
รหัสประเภทที่ลบ	รหัสประเภทที่ผู้ดูแลระบบต้องการลบ	ผู้ดูแลระบบ	Process 3.0 จัดการประเภทวิดีโอ	รหัสประเภท
		Process 3.0 จัดการประเภทวิดีโอ	D3 ข้อมูลประเภทวิดีโอ	
ข้อมูลประเภทที่เพิ่ม	ข้อมูลประเภทที่ผู้ดูแลระบบต้องการเพิ่ม	ผู้ดูแลระบบ	Process 3.0 จัดการประเภทวิดีโอ	รหัสประเภทวิดีโอ + ชื่อประเภทวิดีโอ
		Process 3.0 จัดการประเภทวิดีโอ	D3 ข้อมูลประเภทวิดีโอ	
ข้อมูลประเภทที่แก้ไข	ข้อมูลประเภทที่ผู้ดูแลระบบต้องการแก้ไข	ผู้ดูแลระบบ	Process 3.0 จัดการประเภทวิดีโอ	รหัสประเภทวิดีโอ + ชื่อประเภทวิดีโอ
		Process 3.0 จัดการประเภทวิดีโอ	D3 ข้อมูลประเภทวิดีโอ	
ข้อมูลผู้ใช้ที่จัดการสิทธิ์	ข้อมูลของผู้ใช้ที่ผู้ดูแลระบบต้องการจัดการสิทธิ์	ผู้ดูแลระบบ	Process 4.0 จัดการสิทธิ์การใช้งาน	รหัสผู้ใช้ + สิทธิ์การใช้งาน
		Process 4.0 จัดการสิทธิ์การใช้	D1 ข้อมูลผู้ใช้	

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
รายการข้อมูลผู้ใช้	ข้อมูลของผู้ใช้ที่มีในระบบ	D1 ข้อมูลผู้ใช้	Process 4.0 จัดการสิทธิ์การใช้งาน	{ รหัสผู้ใช้ + ชื่อผู้ใช้ + อีเมล + รหัสผ่าน + ประเภทผู้ใช้ + จำนวนวิดีโอ + ข้อมูลรูปภาพโปรไฟล์ + เวลาที่สร้างบัญชี + สิทธิ์ในการอัปโหลด + แฟ้มข้อมูลในการอัปโหลด + ข้อมูลภาพหน้าปก +พื้นที่หน่วยความจำที่ใช้ }
		Process 4.0 จัดการสิทธิ์การใช้งาน	ผู้ดูแลระบบ	
		D1 ข้อมูลผู้ใช้	Process 5.0 จัดการข้อมูลผู้ใช้	
		Process 5.0 จัดการข้อมูลผู้ใช้	ผู้ดูแลระบบ	
		Process 5.0 จัดการข้อมูลผู้ใช้	ผู้ใช้	
ข้อมูลผู้ใช้ที่เพิ่ม	ข้อมูลผู้ใช้ที่ถูกเพิ่มเข้าระบบ	ผู้ดูแลระบบ	Process 5.0 จัดการข้อมูลผู้ใช้	รหัสผู้ใช้ + ชื่อผู้ใช้ + อีเมล + รหัสผ่าน + ประเภทผู้ใช้ + สิทธิ์ในการอัปโหลด
		Process 5.0 จัดการข้อมูลผู้ใช้	D1 ข้อมูลผู้ใช้	
ข้อมูลผู้ใช้ที่แก้ไข	ข้อมูลของผู้ใช้ที่ต้องการจะแก้ไข	ผู้ใช้	Process 5.0 จัดการข้อมูลผู้ใช้	[ชื่อผู้ใช้ | อีเมล | ข้อมูลรูปภาพโปรไฟล์ | ข้อมูลรูปภาพหน้าปก]
		ผู้ดูแลระบบ	Process 5.0 จัดการข้อมูลผู้ใช้	[ชื่อผู้ใช้ | อีเมล | ประเภทผู้ใช้ | สิทธิ์ในการอัปโหลด]

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
		Process 5.0 จัดการข้อมูลผู้ใช้	D1 ข้อมูลผู้ใช้	รหัสผู้ใช้ + ชื่อผู้ใช้ + อีเมล + รหัสผ่าน + ประเภทผู้ใช้ + ข้อมูลรูปภาพโปรไฟล์ + สิทธิ์ในการอัปโหลด + ข้อมูลภาพหน้าปก
รหัสผู้ใช้ที่ลบ	ข้อมูลของผู้ใช้ที่ต้องการลบ	ผู้ใช้	Process 5.0 จัดการข้อมูลผู้ใช้	รหัสผู้ใช้
		ผู้ดูแลระบบ	Process 5.0 จัดการข้อมูลผู้ใช้	
		Process 5.0 จัดการข้อมูลผู้ใช้	D1 ข้อมูลผู้ใช้	
ข้อมูลวิดีโอที่เพิ่ม	ข้อมูลวิดีโอที่เพิ่มเข้าสู่ระบบ	ผู้ใช้	Process 6.0 จัดการข้อมูลวิดีโอ	ชื่อวิดีโอ + สิทธิ์ของวิดีโอ + (คำอธิบายวิดีโอ) + (ประเภทวิดีโอ)
		ผู้ดูแลระบบ	Process 6.0 จัดการข้อมูลวิดีโอ	
		Process 6.0 จัดการข้อมูลวิดีโอ	D4 ข้อมูลวิดีโอ	รหัสวิดีโอ + ชื่อวิดีโอ + สิทธิ์ของวิดีโอ + (คำอธิบายวิดีโอ) + (ประเภทวิดีโอ)
รหัสวิดีโอที่ลบ	รหัสวิดีโอที่ลบ	ผู้ดูแลระบบ	Process 6.0 จัดการข้อมูลวิดีโอ	รหัสวิดีโอ

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
		ผู้ใช้	Process 6.0 จัดการข้อมูลวิดีโอ	
		Process 6.0 จัดการข้อมูลวิดีโอ	D4 ข้อมูลวิดีโอ	
ข้อมูลวิดีโอที่แก้ไข	ข้อมูลของวิดีโอที่ต้องการแก้ไข	ผู้ดูแลระบบ	Process 6.0 จัดการข้อมูลวิดีโอ	(ชื่อวิดีโอ) + (สิทธิ์ของวิดีโอ) + (คำอธิบายวิดีโอ) + (ประเภทวิดีโอ)
		ผู้ใช้	Process 6.0 จัดการข้อมูลวิดีโอ	
		Process 6.0 จัดการข้อมูลวิดีโอ	D4 ข้อมูลวิดีโอ	
รายการข้อมูลวิดีโอ	ข้อมูลของวิดีโอที่มีในระบบ	D4 ข้อมูลวิดีโอ	Process 6.0 จัดการข้อมูลวิดีโอ	{ รหัสวิดีโอ + ชื่อวิดีโอ + จำนวนการดู + ความยาววิดีโอ + ขนาดของวิดีโอ + เวลาที่ Upload + ข้อมูลภาพตัวอย่าง + รหัสผู้ใช้เจ้าของวิดีโอ + สิทธิ์ของวิดีโอ + ชื่อไฟล์วิดีโอที่ผ่านการแปลง + ขนาดความชัดของวิดีโอ +
		D4 ข้อมูลวิดีโอ	Process 7.0 จัดการข้อมูลการค้นหา	
		Process 6.0 จัดการข้อมูลวิดีโอ	ผู้ใช้	
		Process 6.0 จัดการข้อมูลวิดีโอ	ผู้ดูแลระบบ	
		D4 ข้อมูลวิดีโอ	Process 9.0 ตรวจสอบสถิติ	

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
			การดู	คำอธิบายของวิดีโอ }
		D4 ข้อมูลวิดีโอ	Process 10.0 ตรวจสอบข้อมูลพื้นที่เก็บข้อมูล	
ข้อมูลประเภทของวิดีโอ	ข้อมูลประเภทของวิดีโอที่มีในระบบ	D3 ข้อมูลวิดีโอ	Process 6.0 จัดการข้อมูลวิดีโอ	รหัสประเภทวิดีโอ + ชื่อประเภทวิดีโอ
ข้อมูลที่ค้นหา	ข้อมูลที่ต้องการค้นหาจากระบบ	ผู้ใช้	Process 7.0 จัดการข้อมูลการค้นหา	ข้อความที่ต้องการค้นหา
		ผู้ดูแลระบบ	Process 7.0 จัดการข้อมูลการค้นหา	
รายการข้อมูลการค้นหา	รายการของข้อมูลที่ค้นหาจากระบบ	Process 7.0 จัดการข้อมูลการค้นหา	ผู้ใช้	{ (ข้อมูลวิดีโอ) + (ข้อมูลผู้ใช้) }
		Process 7.0 จัดการข้อมูลการค้นหา	ผู้ดูแลระบบ	
ข้อมูลผู้ใช้ที่ตรวจสอบประวัติการดู	ข้อมูลผู้ใช้ที่ต้องการตรวจสอบประวัติ	ผู้ใช้	Process 8.0 ตรวจสอบประวัติการดู	รหัสผู้ใช้
รายการข้อมูลประวัติการดู	รายการของข้อมูลประวัติจากระบบ	D5 ข้อมูลประวัติการดูวิดีโอ	Process 8.0 ตรวจสอบประวัติการดู	{ รหัสประวัติการดู + รหัสผู้ใช้ + รหัสวิดีโอ + เวลาที่ดูวิดีโอ + เวลาที่ดูค้างไว้ }
		Process 8.0 ตรวจสอบประวัติการดู	ผู้ใช้	
		D5 ข้อมูลประวัติ	Process 9.0	

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

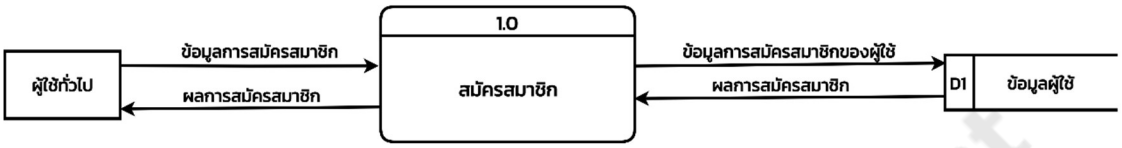
Name	Description	Source	Destination	Data Structure
		การดูวิดีโอ	ตรวจสอบสถิติการดู	
ข้อมูลผู้ใช้ที่ตรวจสอบสถิติการดู	ข้อมูลผู้ใช้ที่ต้องการตรวจสอบสถิติ	ผู้ใช้	Process 9.0 ตรวจสอบสถิติการดู	รหัสผู้ใช้
รายการข้อมูลสถิติการดู	รายการของข้อมูลสถิติจากระบบ	Process 9.0 ตรวจสอบสถิติการดู	ผู้ใช้	{ รหัสวิดีโอ + ชื่อผู้ใช้เจ้าของวิดีโอ + ชื่อวิดีโอ + รูปตัวอย่าง + เวลาที่อัปโหลด + จำนวนการดู }
ข้อมูลผู้ใช้ที่ตรวจสอบพื้นที่เก็บข้อมูล	ข้อมูลผู้ใช้ที่ต้องการตรวจสอบพื้นที่เก็บข้อมูล	ผู้ดูแลระบบ	Process 10.0 ตรวจสอบข้อมูลพื้นที่เก็บข้อมูล	รหัสผู้ใช้
		ผู้ใช้	Process 10.0 ตรวจสอบข้อมูลพื้นที่เก็บข้อมูล	
ข้อมูลพื้นที่เก็บข้อมูล	ข้อมูลพื้นที่เก็บข้อมูล	Process 10.0 ตรวจสอบข้อมูลพื้นที่เก็บข้อมูล	ผู้ใช้	ข้อมูลพื้นที่เก็บข้อมูล
		Process 10.0 ตรวจสอบข้อมูลพื้นที่เก็บข้อมูล	ผู้ดูแลระบบ	
ข้อมูลที่ตรวจสอบ	ข้อมูล Server ที่ต้องการตรวจสอบ	ผู้ดูแลระบบ	Process 11.0 ตรวจสอบข้อมูล Server	ข้อมูล Server
รายการข้อมูล Server	รายการข้อมูลของ Server ที่ตรวจสอบ	เซิร์ฟเวอร์	Process 11.0 ตรวจสอบข้อมูล Server	หน่วยความจำ + การใช้หน่วยประมวลผล +

ตารางที่ 3.3 Data Flow Description and Data Structure of Data Flow (ต่อ)

Name	Description	Source	Destination	Data Structure
		Process 11.0 ตรวจสอบข้อมูล Server	ผู้ดูแลระบบ	พื้นที่เก็บข้อมูล + การเชื่อมต่อกับเครือข่าย
ข้อมูลผู้ใช้ที่ต้องการตรวจสอบการเข้าสู่ระบบ	ข้อมูลผู้ใช้ที่ต้องการตรวจสอบ	ผู้ดูแลระบบ	Process 12.0 ตรวจสอบข้อมูลการเข้าระบบ	รหัสผู้ใช้
		ผู้ใช้	Process 12.0 ตรวจสอบข้อมูลการเข้าระบบ	
รายการข้อมูลประวัติ	ข้อมูลประวัติการเข้าระบบ	D2 ข้อมูลประวัติการเข้าระบบ	Process 12.0 ตรวจสอบข้อมูลการเข้าระบบ	{ รหัสข้อมูล Log + รหัสผู้ใช้ + ประเภทของ Log + เวลาของการเก็บ Log }
		Process 12.0 ตรวจสอบข้อมูลการเข้าระบบ	ผู้ดูแลระบบ	
		Process 12.0 ตรวจสอบข้อมูลการเข้าระบบ	ผู้ใช้	
รายการข้อมูล API ของวิดีโอ	ข้อมูล API ของวิดีโอ	D6 ข้อมูล API ของวิดีโอ	Process 13.0 ตรวจสอบข้อมูล API URL	{ รหัสข้อมูล API + URL Token + เวลาที่สร้าง + เวลาหมดอายุ + รหัสผู้ใช้ + รหัสวิดีโอ }
		Process 13.0 ตรวจสอบข้อมูล API URL	ผู้ดูแลระบบ	
		Process 13.0 ตรวจสอบข้อมูล API URL	ผู้ใช้	

3.6 คำอธิบายการประมวลผล (Process Description)

3.6.1 Process Description 1.0 สมัครสมาชิก

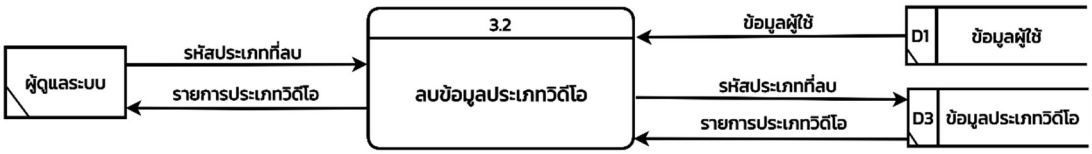


ID	1.0
Name	สมัครสมาชิก
Description	การสมัครเป็นสมาชิกของเว็บไซต์ของผู้ใช้ทั่วไปที่ยังไม่ได้เป็นสมาชิก
Input Data Flow	- ข้อมูลการสมัครสมาชิก - ผลการสมัครสมาชิก
Output Data Flow	- ข้อมูลการสมัครสมาชิกของผู้ใช้ - ผลการสมัครสมาชิก
Process Description	เริ่มต้น 1.รับข้อมูลที่ผู้ใช้กรอกเพื่อสมัครสมาชิก 2.ตรวจสอบข้อมูลการสมัครสมาชิกว่าครบถ้วนและถูกต้องหรือไม่ 2.1 ถ้าข้อมูลถูกต้องและครบถ้วนทุกรายการ ไปที่ข้อ 3. 2.2 ถ้าข้อมูลครบถ้วนแต่ไม่ถูกต้องจะแสดงข้อความ “กรุณาตรวจสอบข้อมูลให้ถูกต้อง” กลับไปที่ข้อ 1. 2.3 ถ้าข้อมูลไม่ครบถ้วนจะแสดงข้อความ “กรุณากรอกข้อมูลให้ครบถ้วน” กลับไปที่ข้อ 1. 3.ตรวจสอบข้อมูลในข้อมูลผู้ใช้ว่ามีข้อมูลอีเมลแล้วหรือไม่ 3.1 ถ้ามีข้อมูลอีเมลอยู่แล้วจะแสดงข้อความ “มี Email นี้แล้ว กรุณาใช้ Email ใหม่แล้วลองอีกครั้ง” กลับไปข้อ 1. 3.2 ถ้าไม่มีข้อมูลอีเมลซ้ำในระบบ ให้เพิ่มข้อมูลการสมัครสมาชิกลงในข้อมูลสมาชิกและแสดงข้อความ “สมัครสมาชิกสำเร็จ” จบการทำงาน

3.6.2 Process Description 2 เข้าสู่ระบบ

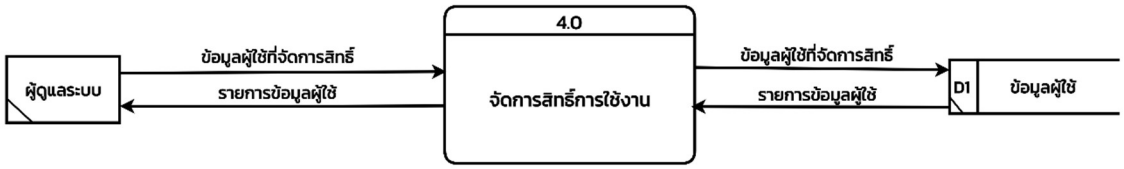
ID	2.0
Name	เข้าสู่ระบบ
Description	ผู้ใช้เข้าสู่ระบบเพื่อใช้งานเว็บไซต์
Input Data Flow	- ข้อมูลการเข้าสู่ระบบ - ข้อมูลผู้ใช้
Output Data Flow	- ข้อมูลการเข้าสู่ระบบ
Process Description	เริ่มต้น 1. รับข้อมูลการเข้าสู่ระบบจากผู้ใช้ (อีเมล และ รหัสผ่าน) 2. ตรวจสอบข้อมูลการเข้าสู่ระบบว่าป้อนข้อมูลได้ครบถ้วนหรือไม่ 2.1 ถ้าป้อนข้อมูลการเข้าสู่ระบบครบถ้วนจะแสดงข้อความ ไปที่ข้อ 3 2.2 ถ้าป้อนข้อมูลการเข้าสู่ระบบไม่ครบถ้วนจะแสดงข้อความ “กรุณาป้อนข้อมูลให้ครบทุกช่อง” กลับไปที่ข้อ 1 3. ตรวจสอบข้อมูลการเข้าสู่ระบบในข้อมูลผู้ใช้ 3.1 ถ้าข้อมูลไม่ตรงกับข้อมูลในข้อมูลผู้ใช้จะแสดงข้อความ “อีเมลหรือรหัสผ่านไม่ถูกต้อง กรุณาลองใหม่อีกครั้ง” กลับไปที่ข้อ 1 3.2 ถ้าข้อมูลตรงกับข้อมูลในข้อมูลผู้ใช้จะแสดงข้อความ “เข้าสู่ระบบสำเร็จ” และบันทึกข้อมูลการเข้าสู่ระบบลงในข้อมูลประวัติการเข้าระบบ จบการทำงาน

3.6.3 Process Description 3.2 ลบข้อมูลประเภทวิดีโอ



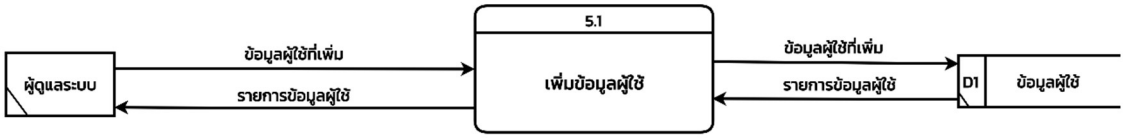
ID	3.2
Name	ลบข้อมูลประเภทวิดีโอ
Description	การลบข้อมูลประเภทวิดีโอออกจากระบบ
Input Data Flow	- รหัสประเภทที่ลบ - ข้อมูลผู้ใช้ - รายการประเภทวิดีโอ
Output Data Flow	- รหัสประเภทที่ลบ - รายการประเภทวิดีโอ
Process Description	เริ่มต้น 1.แสดงรายการข้อมูลประเภทวิดีโอในระบบ 2.เลือกประเภทวิดีโอที่ต้องการลบ 3.แสดงหน้ายืนยันการลบ 3.1 ถ้ายกเลิก ให้กลับไปที่ข้อ 1 3.2 ถ้าตกลง ให้ทำการลบข้อมูลประเภทวิดีโอออกจากข้อมูลประเภทวิดีโอ และแสดงข้อความ “ลบข้อมูลประเภทวิดีโอสำเร็จ” จบการทำงาน

3.6.4 Process Description 4.0 จัดการสิทธิ์การใช้งาน



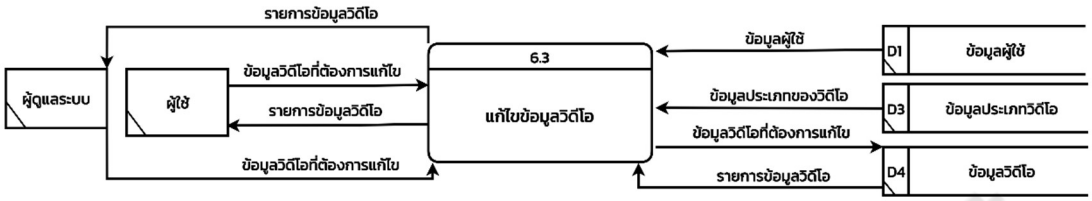
ID	4.0
Name	จัดการสิทธิ์การใช้งาน
Description	การจัดการสิทธิ์การใช้งานผู้ใช้งานในระบบ
Input Data Flow	- ข้อมูลผู้ใช้ที่จัดการสิทธิ์ - รายการข้อมูลผู้ใช้
Output Data Flow	- ข้อมูลผู้ใช้ที่จัดการสิทธิ์ - รายการข้อมูลผู้ใช้
Process Description	เริ่มต้น 1. แสดงรายการข้อมูลผู้ใช้ในระบบ 2. เลือกผู้ใช้ที่ต้องการจัดการสิทธิ์ 3. ตรวจสอบข้อมูลครบถ้วนถูกต้องแล้วหรือไม่ 3.1 ถ้าข้อมูลถูกต้อง ให้บันทึกข้อมูลสิทธิ์การใช้งานลงในข้อมูลผู้ใช้และแสดงข้อความ “จัดการสิทธิ์ผู้ใช้สำเร็จ” 3.2 ถ้าข้อมูลไม่ถูกต้องจะแสดงข้อความ “ข้อมูลไม่ถูกต้อง” กลับไปที่ข้อ 1 จบการทำงาน

3.6.5 Process Description 5.1 เพิ่มข้อมูลผู้ใช้



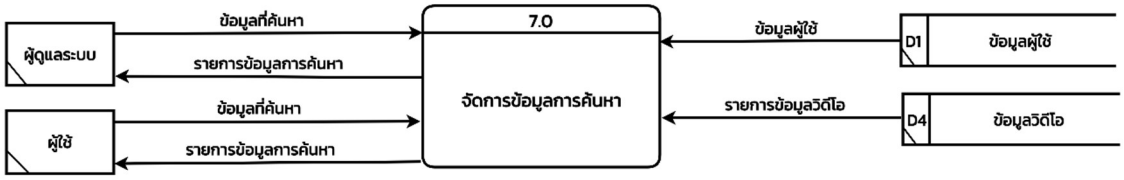
ID	5.1
Name	เพิ่มข้อมูลผู้ใช้
Description	การเพิ่มข้อมูลผู้ใช้
Input Data Flow	- ข้อมูลผู้ใช้ที่เพิ่ม - รายการข้อมูลผู้ใช้
Output Data Flow	- ข้อมูลผู้ใช้ที่เพิ่ม - รายการข้อมูลผู้ใช้
Process Description	เริ่มต้น 1. รับข้อมูลการเพิ่มผู้ใช้ 2. ตรวจสอบข้อมูลการเพิ่มผู้ใช้ครบถ้วนและถูกต้องหรือไม่ 2.1 ถ้าข้อมูลถูกต้องและครบถ้วนทุกรายการ ไปที่ข้อ 3 2.2 ถ้าข้อมูลครบถ้วนแต่ไม่ถูกต้องจะแสดงข้อความ “กรุณาตรวจสอบข้อมูลให้ถูกต้อง” กลับไปที่ข้อ 1 2.3 ถ้าข้อมูลไม่ครบถ้วนจะแสดงข้อความ “กรุณากรอกข้อมูลให้ครบถ้วน” กลับไปที่ข้อ 1 3. ตรวจสอบข้อมูลในข้อมูลผู้ใช้ว่าข้อมูลอีเมลซ้ำหรือไม่ 3.1 ถ้าไม่มีข้อมูลอีเมลซ้ำในระบบ ให้เพิ่มข้อมูลผู้ใช้ลงในข้อมูลผู้ใช้และแสดงข้อความ “เพิ่มผู้ใช้สำเร็จ” 3.2 ถ้ามีข้อมูลอีเมลซ้ำในระบบ แสดงข้อความ “มีข้อมูลอีเมลนี้แล้ว กรุณาใช้ข้อมูลอีเมลใหม่” กลับไปข้อ 1 จบการทำงาน

3.6.6 Process Description 6.3 แก้ไขข้อมูลวิดีโอ



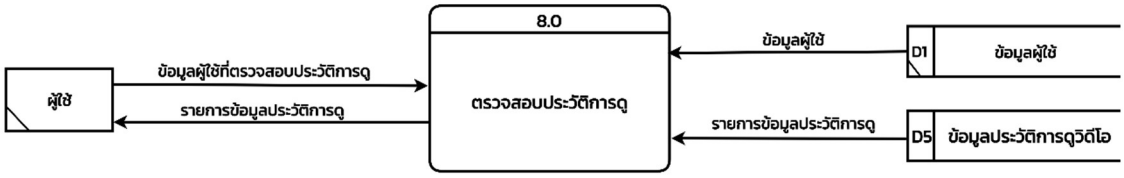
ID	6.3
Name	แก้ไขข้อมูลวิดีโอ
Description	การแก้ไขข้อมูลของวิดีโอที่มีในระบบ
Input Data Flow	- ข้อมูลผู้ใช้ - รายการข้อมูลวิดีโอ - ข้อมูลประเภทของวิดีโอ - ข้อมูลวิดีโอที่ต้องการแก้ไข
Output Data Flow	- รายการข้อมูลวิดีโอ - ข้อมูลวิดีโอที่ต้องการแก้ไข
Process Description	เริ่มต้น 1. ผู้ใช้ป้อนข้อมูลวิดีโอที่ต้องการแก้ไข 2. ตรวจสอบว่าข้อมูลครบถ้วนหรือไม่ 2.1 ถ้าข้อมูลวิดีโอครบถ้วนและถูกต้องทุกรายการให้บันทึกข้อมูลวิดีโอลงในข้อมูลวิดีโอและแสดงข้อความ “แก้ไขข้อมูลวิดีโอสำเร็จ” 2.2 ถ้าข้อมูลวิดีโอไม่ครบถ้วน แสดงข้อความ “กรุณากรอกข้อมูลให้ครบถ้วน” กลับไปข้อ 1 จบการทำงาน

3.6.7 Process Description 7.0 จัดการข้อมูลการค้นหา



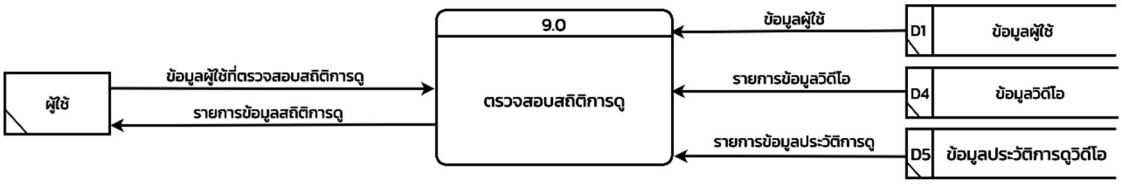
ID	7.0
Name	จัดการข้อมูลการค้นหา
Description	จัดการข้อมูลการค้นหาข้อมูลของผู้ใช้
Input Data Flow	- ข้อมูลที่ค้นหา - ข้อมูลผู้ใช้ - รายการข้อมูลวิดีโอ
Output Data Flow	- รายการข้อมูลการค้นหา
Process Description	เริ่มต้น 1. รับข้อมูลจากผู้ใช้ที่ต้องการค้นหา 2. ค้นหาข้อมูลผู้ใช้และข้อมูลวิดีโอตามข้อมูลที่ค้นหา 3. ตรวจสอบว่ามีข้อมูลหรือไม่ 3.1 ถ้ามีข้อมูล แสดงรายการข้อมูลตามที่ผู้ใช้ค้นหา 3.2 ถ้าไม่มีข้อมูลให้แสดงข้อความ “ไม่มีข้อมูลที่ค้นหา” แล้วกลับไปข้อ 1 จบการทำงาน

3.6.8 Process Description 8.0 ตรวจสอบประวัติการดู



ID	8.0
Name	ตรวจสอบประวัติการดู
Description	ตรวจสอบประวัติการดูของผู้ใช้
Input Data Flow	- ข้อมูลผู้ใช้ที่ตรวจสอบประวัติการดู - ข้อมูลผู้ใช้ - รายการข้อมูลประวัติการดู
Output Data Flow	- รายการข้อมูลประวัติการดู
Process Description	เริ่มต้น 1. รับข้อมูลผู้ใช้ที่ต้องการตรวจสอบ 2. ค้นหาข้อมูลประวัติการดูจากข้อมูลประวัติการดูวิดีโอตามข้อมูลผู้ใช้ 3. ตรวจสอบว่ามีข้อมูลหรือไม่ 3.1 ถ้ามีข้อมูล แสดงรายการข้อมูลประวัติการดูวิดีโอ 3.2 ถ้าไม่มีข้อมูลให้แสดงข้อความ “ไม่มีประวัติการดูวิดีโอ” กลับไปข้อ 1 จบการทำงาน

3.6.9 Process Description 9.0 ตรวจสอบสถิติการดู

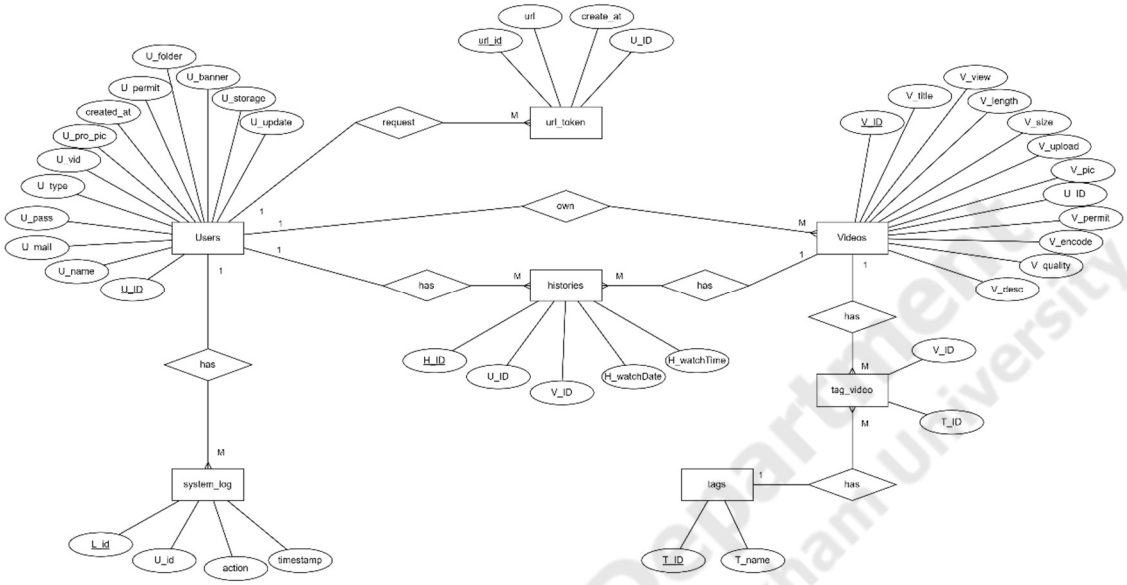


ID	9.0
Name	ตรวจสอบสถิติการดู
Description	การตรวจสอบสถิติการดูของผู้ใช้
Input Data Flow	- ข้อมูลผู้ใช้ที่ตรวจสอบสถิติการดู - รายการข้อมูลวิดีโอ - ข้อมูลผู้ใช้ - รายการข้อมูลประวัติการดู
Output Data Flow	- รายการข้อมูลสถิติการดู
Process Description	เริ่มต้น 1. รับข้อมูลผู้ใช้ที่ต้องการตรวจสอบ 2. ค้นหาข้อมูลสถิติการดูจากข้อมูลประวัติการดูวิดีโอและข้อมูลวิดีโอตามข้อมูลผู้ใช้ 3. ตรวจสอบว่ามีข้อมูลหรือไม่ 3.1 ถ้ามีข้อมูล แสดงรายการข้อมูลสถิติการดูวิดีโอ 3.2 ถ้าไม่มีข้อมูลให้แสดงข้อความ “ไม่มีสถิติการดูวิดีโอ” กลับไปข้อ 1 จบการทำงาน

3.6.10 Process Description 13.0 ตรวจสอบข้อมูล API URL

ID	13.0
Name	ตรวจสอบข้อมูล API URL
Description	การตรวจสอบข้อมูล API URL ของผู้ใช้
Input Data Flow	- รายการข้อมูล API ของวิดีโอ - ข้อมูลผู้ใช้
Output Data Flow	- รายการข้อมูล API ของวิดีโอ
Process Description	เริ่มต้น 1. รับข้อมูลผู้ใช้ที่ต้องการตรวจสอบ 2. ค้นหาข้อมูล API จากข้อมูล API ของวิดีโอตามข้อมูลผู้ใช้ 3. ตรวจสอบว่ามีข้อมูลหรือไม่ 3.1 ถ้ามีข้อมูล แสดงรายการข้อมูล API ของวิดีโอ 3.2 ถ้าไม่มีข้อมูล ให้แสดงข้อความ “ไม่มีข้อมูล API” แล้วกลับไปข้อ 1 จบการทำงาน

3.7 แผนภาพความสัมพันธ์ (Entity Relationship Diagram)



ภาพประกอบที่ 3.11 Entity Relationship Diagram ของระบบ

3.8 การออกแบบฐานข้อมูล (Database Design)

ตารางที่ 3.4 ตารางผู้ใช้ (users)

Id	Column	Type	Description	Example Data	Constraint
1	U_ID	int(11)	รหัสผู้ใช้	1	PK
2	U_name	varchar(255)	ชื่อผู้ใช้	John	Not Null
3	U_mail	varchar(50)	เมลผู้ใช้	john@mail.com	Not Null
4	U_pass	varchar(30)	รหัสผ่านผู้ใช้	ec457d0a974c48d5685a7efa03d137dc8bbde7e3	Not Null
5	U_type	varchar (5)	ประเภทผู้ใช้ (user, admin)	admin	can be only user and admin
6	U_vid	int(11)	จำนวนวิดีโอที่ผู้ใช้เป็นเจ้าของ	2	Not Null
7	U_pro_pic	longblob	ข้อมูล base64	iVBORw0KGgoAA	Not Null

ตารางที่ 3.4 ตารางผู้ใช้ (users) (ต่อ)

Id	Column	Type	Description	Example Data	Constraint
			สำหรับภาพโปรไฟล์	AANSUhEUg==	
8	created_at	timestamp	เวลาที่บัญชีถูกสร้าง	1/17/2023 15:53	DEFAULT_GENERATED
9	U_permit	tinyint(1)	สิทธิ์ในการอัปโหลดวิดีโอ (0=ไม่สามารถอัปโหลดได้, 1=สามารถอัปโหลดได้)	1	can be only 0 and 1
10	U_folder	varchar(20)	ชื่อโฟลเดอร์ที่ทำการเก็บข้อมูลวิดีโอ	I7J5dDzb2hSL	Not Null
11	U_banner	longblob	ข้อมูล base64 สำหรับหน้าปกของผู้ใช้	iVBORw0KGgoAAAANSUhEUg==	Not Null
12	U_storage	int(11)	ข้อมูลการใช้พื้นที่การเก็บข้อมูลจากขนาดวิดีโอ (MB)	132	0
13	U_update	tinyint(1)	ค่าสำหรับการตรวจสอบว่ามีการเปลี่ยนแปลงของข้อมูลหรือไม่ (0=ไม่มีการเปลี่ยนแปลง, 1=มีการเปลี่ยนแปลง)	1	can be only 0 and 1

ตารางที่ 3.5 ตารางวิดีโอ (videos)

Id	Column	Type	Description	Example Data	Constraint
1	V_ID	int(11)	รหัสวิดีโอ	1	PK
2	V_tilte	varchar(255)	ชื่อวิดีโอ	How to PHP	Not Null
3	V_view	int(11)	จำนวนการดูวิดีโอ	23	Not Null
4	V_length	varchar(50)	ความยาววิดีโอ	5:12	Not Null
5	V_size	int(11)	ขนาดของวิดีโอ	33793765	Not Null
6	V_upload	timestamp	เวลาที่อัปโหลด	2/16/2023 11:01	DEFAULT_GENERATED
7	V_pic	longblob	ข้อมูล base64 สำหรับภาพตัวอย่าง	iVBORw0KGgoAAAANSUhEUg	Not Null
8	U_ID	int(11)	รหัสของเจ้าของวิดีโอ	1	FK: reference from users(U_ID)
9	V_permit	varchar(50)	ประเภทของวิดีโอ (public, private, unlisted)	public	Not Null
10	V_encode	varchar(50)	ชื่อไฟล์วิดีโอที่ผ่านการแปลงชื่อและเก็บไว้ใน server แล้ว	63ee0cfb7fea45_10534973	Not Null
11	V_quality	varchar(10)	ขนาดความละเอียดสูงสุด	720	Not Null

ตารางที่ 3.5 ตารางวิดีโอ (videos) (ต่อ)

Id	Column	Type	Description	Example Data	Constraint
			ของวิดีโอนั้น ๆ		
12	V_desc	varchar(255)	คำอธิบายวิดีโอนั้น ๆ	“This is sample video”	Null

ตารางที่ 3.6 ตารางเก็บข้อมูลระบบ (system_logs)

Id	Column	Type	Description	Example Data	Constraint
1	L_id	int(10)	รหัสข้อมูล log	1	PK
2	U_id	int(11)	รหัสผู้ใช้	1	FK: reference from users(U_ID)
3	action	Varchar(10)	ประเภทของ log ที่เกิดขึ้น	login	Not Null
4	create_at	timestamp	เวลาที่มีการเก็บ log	1/27/2023 19:30	DEFAULT_GENERATED

ตารางที่ 3.7 ตารางเก็บข้อมูลประวัติการเข้าชม (histories)

Id	Column	Type	Description	Example Data	Constraint
1	H_ID	int(20)	รหัสประวัติการดู	1	PK
2	U_ID	int(20)	รหัสผู้ใช้	2	FK: reference from users(U_ID)
3	V_ID	int(20)	รหัสวิดีโอ	12	FK: reference from

ตารางที่ 3.7 ตารางเก็บข้อมูลประวัติการเข้าชม (histories) (ต่อ)

Id	Column	Type	Description	Example Data	Constraint
					videos(V_ID)
4	H_watchDate	timestamp	เวลาที่ดูวิดีโอ	2/26/2023 23:07	DEFAULT_GENERATED
5	H_watchTime	Float	เวลาในวิดีโอที่ดูค้างไว้	12.2016	Not Null

ตารางที่ 3.8 ตารางข้อมูลประเภทวิดีโอ (tags)

Id	Column	Type	Description	Example Data	Constraint
1	T_ID	int(11)	รหัสประเภทวิดีโอ	1	PK
2	T_name	char(255)	ชื่อประเภท	Coding	Not Null

ตารางที่ 3.9 ตารางวิดีโอและประเภท (tag_video)

Id	Column	Type	Description	Example Data	Constraint
1	V_ID	int(11)	รหัสวิดีโอ	1	FK: reference from videos(V_ID)
2	T_ID	int(11)	รหัสประเภทวิดีโอ	1	FK: reference from tags(T_ID)

ตารางที่ 3.10 ตาราง url สำหรับการเข้าถึงวิดีโอ (url_token)

Id	Column	Type	Description	Example Data	Constraint
1	url_id	int(10)	รหัสข้อมูล url	1	PK

ตารางที่ 3.10 ตาราง url สำหรับการเข้าถึงวิดีโอ (url_token) (ต่อ)

Id	Column	Type	Description	Example Data	Constraint
2	url	varchar(30)	url สำหรับการเข้าถึงไฟล์วิดีโอ	513988dc6324c57f6d265543b4b8ce19	Not Null
3	create_at	Timestamp	เวลาที่ url ถูกสร้างขึ้นมา	2/26/2023 23:07	current_timestamp()
4	U_ID	int(11)	รหัสของผู้ใช้ที่ขอ request	1	FK: reference from users(U_ID)

3.9 การพัฒนาระบบ

3.9.1 โค้ดส่วนการส่งข้อมูลทรัพยากรของเครื่อง

- ใช้ flask python ในการสร้าง api สำหรับการตรวจสอบเครื่อง
- ใช้ library psutil ในการตรวจสอบข้อมูลทรัพยากรเครื่อง
- บรรทัดที่ 12 เป็นการนำข้อมูลการใช้หน่วยประมวลผลออกมาเก็บไว้ในตัวแปร cpu_used
- บรรทัดที่ 15 เป็นการดึงข้อมูลหน่วยความจำออกมาเก็บไว้ในตัวแปร mem
- บรรทัดที่ 18-22 เป็นการคำนวณปริมาณของพื้นที่เก็บข้อมูลของเครื่องเซิร์ฟเวอร์
- บรรทัดที่ 25-35 เป็นการตรวจสอบข้อมูลการดาวโหลดและอัปโหลดของเครือข่าย
- บรรทัดที่ 37-44 เป็นการทำให้ข้อมูลทั้งหมดอยู่ในรูปแบบ json สำหรับการเรียกข้อมูลผ่าน API

```python
        cpu_used = psutil.cpu_percent()

        #Memory INFO [Ram Used]
        mem = psutil.virtual_memory().percent

        #Disk INFO
        disk_usage = psutil.disk_usage(os.getcwd())
        disk_total = round(disk_usage.total / (1024*1024*1024), 2)
        disk_used = round(disk_usage.used / (1024*1024*1024), 2)
        disk_free = round(disk_usage.free / (1024*1024*1024), 2)
        disk_used_percent = round(disk_usage.percent,2)

        #Network INFO
        inf = "Ethernet"
        net_stat = psutil.net_io_counters(pernic=True, nowrap=True)[inf]
        net_in_1 = net_stat.bytes_recv
        net_out_1 = net_stat.bytes_sent
        time.sleep(1)
        net_stat = psutil.net_io_counters(pernic=True, nowrap=True)[inf]
        net_in_2 = net_stat.bytes_recv
        net_out_2 = net_stat.bytes_sent

        net_in = round((net_in_2 - net_in_1) / 1024 / 1024, 3)
        net_out = round((net_out_2 - net_out_1) / 1024 / 1024, 3)

        return jsonify({'CPU_Used': str(cpu_used)+" %",
                        'Memory_Used' : str(mem)+" %",
                        'Disk_Total' : str(disk_total)+' GB',
                        'Disk_Used' : str(disk_used)+' GB',
                        'Disk_Free' : str(disk_free)+' GB',
                        'Disk_Used_Percent' : str(disk_used_percent)+' %',
                        'Network_Download' : str(net_in)+' MB/s',
                        'Network_Upload' : str(net_out)+' MB/s'})
    return app
```

ภาพประกอบที่ 3.12 ภาพประกอบโค้ดส่วนการตรวจสอบทรัพยากรเครื่องเซิร์ฟเวอร์

```
// 20230323144544
// http://localhost:8900/server_resource/

{
  "CPU_Used": "58.0 %",
  "Disk_Free": "299.34 GB",
  "Disk_Total": "500.0 GB",
  "Disk_Used": "200.66 GB",
  "Disk_Used_Percent": "40.1 %",
  "Memory_Used": "76.1 %",
  "Network_Download": "0.0 MB/s",
  "Network_Upload": "0.0 MB/s"
}
```

ภาพประกอบที่ 3.13 ภาพตัวอย่างการเรียกใช้ API flask python

3.9.2 โค้ดส่วนสมัครสมาชิก

- บรรทัดที่ 288 รับค่าข้อมูลผู้ใช้เข้ามาเก็บไว้ในตัวแปร data
- บรรทัดที่ 297 ทำการ hash password เพื่อเก็บข้อมูลลงไปยังฐานข้อมูล
- บรรทัดที่ 301 – 304 ทำการบันทึกข้อมูลลงไปยังฐานข้อมูล
- บรรทัดที่ 309 - 310 ทำการสร้างโฟลเดอร์สำหรับเก็บข้อมูลวิดีโอของผู้ใช้คนนั้น ๆ บนเซิร์ฟเวอร์

```
288         data = request.get_json()
289         # create connection
290         conn = create_conn()
291
292         username = data.get("username")
293         email = data.get("email")
294         password = data.get("password")
295
296         encode_password = str(password).encode("utf-8")
297         hashed_password = bcrypt.hashpw(encode_password, bcrypt.gensalt())
298         file_name = genFileName(username)
299
300         cursor = conn.cursor()
301         cursor.execute(
302             "INSERT INTO users (U_name, U_mail, U_pass, U_type, U_vid, U_permit, U_folder)
303             (username, email, hashed_password, "user", 0, 1, file_name),
304         )
305         conn.commit()
306         cursor.close()
307         conn.close()
308
309         folder_path = "../upload/" + file_name  # create folder for uploaded videos
310         os.makedirs(folder_path)
311         return ({"status": "success", "data": {}}), 200
```

ภาพประกอบที่ 3.14 ภาพประกอบโค้ดส่วนการสมัครสมาชิก

3.9.3 โค้ดส่วนเข้าสู่ระบบ

- บรรทัดที่ 317 รับข้อมูลผู้ใช้เข้ามา
- บรรทัดที่ 323 รับค่าว่าผู้ใช้กด remember me ไว้หรือไม่
- บรรทัดที่ 327 ดึงข้อมูลจากฐานข้อมูลตาม email ที่ผู้ใช้ใส่เข้ามา

```
317         data = request.get_json()
318
319         conn = create_conn()
320
321         email = data.get("email")
322         plain_password = data.get("password")
323         check = data.get("check")
324
325         # get password
326         cursor = conn.cursor()
327         cursor.execute("SELECT * FROM users WHERE U_mail=%s", (email,))
328         # print('select success')
329         data = cursor.fetchone()
330         # print('get hashed')
331
```

ภาพประกอบที่ 3.15 ภาพประกอบโค้ดส่วนการเข้าสู่ระบบ

- บรรทัดที่ 336 ทำการตรวจสอบว่า password ที่ผู้ใช้กรอกเข้ามา (plain_password) นั้นมีค่า hash ตรงกับค่าที่อยู่ในฐานข้อมูลหรือไม่
- บรรทัดที่ 343 ตรวจสอบว่าผู้ใช้กด remember me ไว้หรือไม่
- บรรทัดที่ 345 และ 353 เป็นการกำหนดค่า jwt และตั้งเวลาหมดอายุตามที่ผู้ใช้กด remember me ไว้หรือไม่

```
333         if data is not None:
334             hashed_password = data[3]
335
336             if bcrypt.checkpw(
337                 plain_password.encode("utf-8"), hashed_password.encode("utf-8")
338             ):
339                 conn.commit()
340                 cursor.close()
341                 conn.close()
342
343                 if check == 1 or check == "1":
344                     print("1 month")
345                     payload = {
346                         "U_id": data[0],
347                         "U_type": data[4],
348                         "U_permit": data[8],
349                         "exp": datetime.utcnow() + timedelta(days=30),
350                     }
351                 else:
352                     print("1 hours")
353                     payload = {
354                         "U_id": data[0],
355                         "U_type": data[4],
356                         "U_permit": data[8],
357                         "exp": datetime.utcnow() + timedelta(days=1),
```

ภาพประกอบที่ 3.15 ภาพประกอบโค้ดส่วนการเข้าสู่ระบบ (ต่อ)

- บรรทัดที่ 360 เป็นการ encode ข้อมูลลงไปยัง jwt
- บรรทัดที่ 362 เป็นการส่งค่าต่าง ๆ ที่จำเป็นต้องใช้ในหน้าเว็บกลับไปยังผู้ใช้

```
360             token = jwt.encode(payload, app.config["SECRET_KEY"], algorithm="HS256")
361             tmp = str(data[6])
362             return (
363                 {
364                     "status": "success",
365                     "token": token,
366                     "data": {
367                         "U_id": data[0],
368                         "username": data[1],
369                         "email": data[2],
370                         "U_type": data[4],
371                         "vid": data[5],
372                         "U_pro_pic": tmp[2:-1],
373                         "U_permit": data[8],
374                         "U_folder": data[9],
375                     },
376                 }
377             ), 200
```

ภาพประกอบที่ 3.15 ภาพประกอบโค้ดส่วนการเข้าสู่ระบบ (ต่อ)

3.9.4 โค้ดส่วนการเรียกดูข้อมูลวิดีโอ

- บรรทัดที่ 922 ทำการรับค่า v จาก url ซึ่งเป็นชื่อวิดีโอ
- บรรทัดที่ 923 ทำการรับค่า u จาก url ซึ่งเป็นชื่อผู้ใช้
- บรรทัดที่ 928 เป็นการดึงข้อมูลวิดีโอต่าง ๆ ทั้งข้อมูลวิดีโอและข้อมูลเจ้าของวิดีโอ

```
922         V_encode = request.args.get("v")
923         u = request.args.get("u")
924
925         conn = create_conn()
926
927         cursor = conn.cursor()
928         cursor.execute(
929             "SELECT v.*, u.U_name, u.U_folder, u.U_pro_pic, COALESCE(h.H_watchTime, 0) AS H_watchTime \
930             FROM videos AS v  \
931             JOIN users AS u ON u.U_ID = v.U_ID \
932             LEFT JOIN histories AS h ON h.U_ID = %s AND h.V_ID = v.V_ID \
933             WHERE u.U_ID = v.U_ID \
934             AND v.V_encode = %s \
935             ORDER BY h.H_watchdata DESC LIMIT 1",
936             (
937                 u,
938                 V_encode,
939             ),
940         )
941         data = cursor.fetchone()
942
```

ภาพประกอบที่ 3.16 ภาพประกอบโค้ดส่วนการเรียกดูข้อมูลวิดีโอ

- บรรทัดที่ 943 เป็นการดึงข้อมูล tag ของวิดีโอนั้น ๆ จากฐานข้อมูล

- บรรทัดที่ 956 จะเป็นการวนเก็บค่าข้อมูล tag ลงไปในตัวแปร tags เพื่อนำไปแสดงในหน้าเว็บ

```
943             cursor.execute(
944                 "SELECT * FROM tags WHERE T_ID IN \
945                     (SELECT T_ID FROM tag_video WHERE V_ID = \
946                         (SELECT V_ID FROM videos WHERE V_encode = %s))",
947                 (V_encode,),
948             )
949             data2 = cursor.fetchall()
950 
951             conn.commit()
952             cursor.close()
953             conn.close()
954 
955             tags = []
956             for row in data2:
957                 tag = {"T_ID": row[0], "T_name": row[1]}
958                 tags.append(tag)
```

ภาพประกอบที่ 3.16 ภาพประกอบโค้ดส่วนการเรียกดูข้อมูลวิดีโอ (ต่อ)

- บรรทัดที่ 963 เป็นการกำหนดค่าที่จะต้องส่งคืนไปยังผู้ใช้
- บรรทัดที่ 982 ทำการส่งค่าคืนผู้ใช้

```
960             tmp = str(data[6])
961             tmp2 = str(data[14])
962             video = {
963                 "V_ID": data[0],
964                 "V_title": data[1],
965                 "V_view": data[2],
966                 "V_length": data[3],
967                 "V_size": data[4],
968                 "V_upload": data[5],
969                 "V_pic": tmp[2:-1],
970                 "U_ID": data[7],
971                 "V_permit": data[8],
972                 "V_encode": data[9],
973                 "V_quality": data[10],
974                 "V_desc": data[11],
975                 "U_name": data[12],
976                 "U_folder": data[13],
977                 "U_pro_pic": tmp2[2:-1],
978                 "watchTime": data[15],
979                 "tags": tags,
980             }
981 
982             return jsonify(video), 200
```

ภาพประกอบที่ 3.16 ภาพประกอบโค้ดส่วนการเรียกดูข้อมูลวิดีโอ (ต่อ)

3.9.5 โค้ดส่วนการเล่นวิดีโอ

- บรรทัดที่ 37 จะเป็นการกำหนด url ที่จะใช้ในการเรียกไฟล์วิดีโอ
- บรรทัดที่ 40 – 48 เป็นการ featch api แบบ post ไปยัง http://domain.com /get/dynamicUrl/token เพื่อให้ได้ url สำหรับเข้าถึงไฟล์วิดีโอ
- บรรทัดที่ 50 จะทำการเก็บค่า response จาก api มาเพื่อส่งต่อไปยัง player บนหน้าเว็บ

```
34      useEffect(() => {
35
36          const tmp = {
37              'vid_url': '/watch?u=' + user + '&v=' + video
38          }
39          console.log(getToken());
40          fetch(dynamicAPI, {
41              method: 'POST',
42              headers: {
43                  'Content-Type': 'application/json',
44                  'Authorization': 'Bearer ' + getToken()
45              },
46              body: JSON.stringify(tmp)
47
48          }).then(response => response.json())
49          .then(data => {
50              setUrl(data.url)
51          })
52          .catch(() => {})
53      }, [dynamicAPI])
```

ภาพประกอบที่ 3.17 ภาพประกอบโค้ดส่วนการเล่นวิดีโอ

- โดยการทำงานของ api ที่ถูกเรียกนั้นเป็นดังนี้
- บรรทัดที่ 1578 จะทำการรับ url วิดีโอที่ผู้ใช้ใส่เข้ามา
- บรรทัดที่ 1579 จะทำการสุ่มค่าฐาน 16 จำนวน 16 บิตมา
- บรรทัดที่ 1580 จะทำการกำหนดเวลาหมดอายุของ url ตัวนี้(1 วัน)
- บรรทัดที่ 1585 ทำการบันทึก ค่าฐาน 16 ที่สุ่มออกมาลงไปยังฐานข้อมูล
- บรรทัดที่ 1599 ทำการสร้าง dynamic url สำหรับการเข้าถึงไฟล์วิดีโอ

```
1576    def getDynamicWToken():
1577        data = request.get_json()
1578        vid_url = data.get('vid_url')
1579        url_token = secrets.token_hex(16)
1580        expiration_time = time.time() + 86400
1581
1582        conn = create_conn()
1583        cursor = conn.cursor()
1584
1585        cursor.execute(
1586                'INSERT INTO url_token(url, url_expire) VALUES (%s, %s)',
1587                (url_token, expiration_time)
1588            )
1589
1590        conn.commit()
1591        cursor.close()
1592        conn.close()
1593
1594        vid_url = vid_url.split('watch?')
1595        path = vid_url[-1].split('&')
1596        u = path[0][2:]
1597        v = path[1][2:]
1598
1599        dynamic_url = f"http://localhost:8900/get/hls/{url_token}/{u}/{v}"
```

ภาพประกอบที่ 3.17 ภาพประกอบโค้ดส่วนการเล่นวิดีโอ (ต่อ)

- หลังจากที่ได้ url มาแล้วทำการนำไปใส่ใน source ของ videoPlayer เพื่อให้เล่นวิดีโอได้

```
<div className='row' style={{paddingBottom: '10px'}}>
    <VideoPlayer source={url} V_id={vidDetail.V_ID} watchTime={vidDetail.watchTime}/>
</div>
```

ภาพประกอบที่ 3.17 ภาพประกอบโค้ดส่วนการเล่นวิดีโอ (ต่อ)

- เมื่อ dynamic url ถูกเรียกใช้
- บรรทัดที่ 1608 จะทำการดึงค่าเวลาหมดอายุของ url จากค่าที่เป็นการสุ่มฐาน 16 จากฐานข้อมูลออกมา
- บรรทัดที่ 1614 จะเป็นการตรวจสอบว่าเวลาของ url นั้นหมดอายุหรือยัง
- หากไม่หมดอายุจะทำการส่งไฟล์วิดีโอกลับไปยังเครื่องเล่นวิดีโอ

```
1603    @app.route('/get/hls/<path:url_token>/<path:u>/<path:v>')
1604    def getHls(url_token, u, v):
1605        conn = create_conn()
1606        cursor = conn.cursor()
1607
1608        cursor.execute(
1609                'SELECT url_expire FROM url_token WHERE url = %s',
1610                (url_token,)
1611            )
1612        data = cursor.fetchone()
1613
1614        if(time.time() <= data[0]):
```

ภาพประกอบที่ 3.17 ภาพประกอบโค้ดส่วนการเล่นวิดีโอ (ต่อ)

3.9.6 โค้ดส่วนเก็บประวัติการรับชม

- บรรทัดที่ 1202 ทำการรับค่า json ที่ส่งมา
- บรรทัดที่ 1206 ทำการ insert ค่าข้อมูลต่าง ๆ สำหรับการบันทึกข้อมูลประวัติการเข้าชมเข้าไปยังฐานข้อมูล

```
1201    def insertHistory():
1202        data = request.get_json()
1203        conn = create_conn()
1204        cursor = conn.cursor()
1205
1206        cursor.execute(
1207            "INSERT INTO histories (U_ID, V_ID, H_watchtime) \
1208                SELECT %s, %s, COALESCE((SELECT H_watchTime FROM histories WHERE U_ID =
1209                WHERE %s != ( \
1210                    SELECT V_ID FROM histories \
1211                    WHERE U_ID = %s \
1212                    ORDER BY H_watchData DESC LIMIT 1) \
1213                    OR NOT EXISTS ( \
1214                        SELECT V_ID FROM histories WHERE U_ID = %s)",
1215            (
1216                data["U_id"],
1217                data["V_id"],
1218                data["U_id"],
1219                data["V_id"],
1220                data["V_id"],
1221                data["U_id"],
1222                data["U_id"],
1223            ),
1224        )
```

ภาพประกอบที่ 3.18 ภาพประกอบโค้ดส่วนการเก็บประวัติการรับชม

3.9.7 โค้ดส่วนอัปโหลดวิดีโอ

- บรรทัดที่ 164 ทำการตรวจสอบว่ามีไฟล์อัปโหลดเข้ามาหรือไม่
- บรรทัดที่ 169 ทำการสร้างตัวแปลสำหรับเก็บข้อมูลวิดีโอที่อัปโหลดเข้ามา
- บรรทัดที่ 171 ทำการดึงข้อมูลจากวิดีโอผ่าน file โดยอ้างอิงจาก path
- บรรทัดที่ 173 ทำการเพิ่ม event listener เพื่อกำหนดค่าต่าง ๆ ไปยังตัวแปรที่จะทำการเก็บข้อมูลลงไปยังฐานข้อมูล
- บรรทัดที่ 196 ทำการสร้างตัวแปร formdata สำหรับส่งผ่าน api โดยมีข้อมูลต่าง ๆ ของวิดีโอและไฟล์วิดีโอ

```
161     const handleClickUpload = async () => {
162
163         setUpProgress('0');
164         if (file) {
165
166             document.getElementById('submitBtn').disabled = true;
167             document.getElementById('cancleBtn').disabled = true;
168
169             const video = document.createElement('video');
170             video.preload = 'metadata';
171             video.src = URL.createObjectURL(file);
```

ภาพประกอบที่ 3.19 ภาพประกอบโค้ดส่วนการอัปโหลดวิดีโอ

```
173             video.addEventListener('loadedmetadata', function () {
174                 const duration = video.duration;
175                 const w = video.videoWidth;
176                 const h = video.videoHeight;
177                 const type = file.type.split('/').pop(); // video/m
178                 const tmpData = {
179                     'videoName': tmp,
180                     'videoOriginName': file.name,
181                     'videoSize': file.size,
182                     'videoDuration': duration,
183                     'videoDesc': vidDesc,
184                     'videoType': type,
185                     'videoOwner': session.U_id,
186                     'videoPermit': vidPermit,
187                     'videoThumbnail': thumbnail,
188                     'path': session.U_folder,
189                     'width': w,
190                     'height': h,
191                     'tags': vidTags
192                     // encode included
193                 }
```

ภาพประกอบที่ 3.19 ภาพประกอบโค้ดส่วนการอัปโหลดวิดีโอ (ต่อ)

```
196             const formData = new FormData();
197             formData.append('video', file, file.name);
198             formData.append('data', JSON.stringify(tmpData));
199             // console.log(tmpData);
200             axios.post(uploadAPI, formData, {
201                 headers: {
202                     'Content-Type': 'multipart/form-data',
203                     'Authorization': 'Bearer ' + token
204                 },
```

ภาพประกอบที่ 3.19 ภาพประกอบโค้ดส่วนการอัปโหลดวิดีโอ (ต่อ)

3.9.8 โค้ดส่วนการแปลงวิดีโอและเก็บข้อมูลวิดีโอ

- บรรทัดที่ 285 ทำการยืนยัน jwt token ว่าถูกต้องหรือไม่
- บรรทัดที่ 287 ทำการเอาค่าต่าง ๆ ออกมาจาก jwt

```
281 @app.route('/upload', methods=['POST'])
282 def upload():
283     token = request.headers.get('Authorization')
284
285     if(verify(token)):
286         tmp = token.split(' ')[-1]
287         payload = jwt.decode(tmp, app.config['SECRET_KEY'], algorithms=['HS256'])
```

ภาพประกอบที่ 3.20 ภาพประกอบโค้ดส่วนการแปลงวิดีโอและเก็บข้อมูลวิดีโอ

- บรรทัดที่ 289 ทำการตรวจสอบว่าผู้ใช้มีสิทธิ์ในการอัปโหลดวิดีโอหรือไม่
- บรรทัดที่ 290 และ 291 ทำการรับข้อมูลไฟล์วิดีโอและข้อมูลของวิดีโอมาตามลำดับ
- บรรทัดที่ 295 ทำการสร้างชื่อใหม่ให้วิดีโอเพื่อไม่ให้เกิดการซ้ำของชื่อในฐานข้อมูล
- บรรทัดที่ 303 บันทึกไฟล์ที่ผู้ใช้อัปโหลดไปยังโฟลเดอร์ที่ระบบเตรียมไว้ให้เพื่อเตรียมสำหรับการแปลงวิดีโอ
- บรรทัดที่ 306 จะเป็นการรอจนกว่าไฟล์จะอัปโหลดไปยังเซิร์ฟเวอร์เสร็จสิ้น
- บรรทัดที่ 309 จะเป็นการสร้าง thread ไว้สำหรับการแปลงวิดีโอ

```
289             if(payload.get('U_permit') == 1):
290                 file = request.files['video']
291                 data = request.form['data']
292 
293                 vid_data = json.loads(data)
294                 vidName = file.filename
295                 new_name = genFileName(vidName.split('.')[0])
296 
297                 vid_data['encode'] = new_name
298 
299                 # print(vid_data)
300 
301                 if file:
302                     save_path = '../upload/'+ vid_data['path'] +'/'+ new_name + '.' + vid_data['videoType']
303                     file.save(save_path)
304 
305                     # Wait until the file is successfully saved
306                     while not os.path.exists(save_path):
307                         time.sleep(1)
```

ภาพประกอบที่ 3.20 ภาพประกอบโค้ดส่วนการแปลงวิดีโอและเก็บข้อมูลวิดีโอ (ต่อ)

```
309                     thread = threading.Thread(target=convert, args=(save_path, vid_data))
310                     thread.start()
311                     # convert(save_path, vid_data)
312                     time.sleep(0.8)
313                     return ({'message': 'upload success, converting'}), 200
```

ภาพประกอบที่ 3.20 ภาพประกอบโค้ดส่วนการแปลงวิดีโอและเก็บข้อมูลวิดีโอ (ต่อ)

- บรรทัดที่ 159 จะทำการตรวจสอบว่าผู้ใช้ได้ใส่ภาพหน้าปกวิดีโอมาด้วยหรือไม่ หากไม่ได้ใส่มาจะทำการไปดึงภาพจากวิดีโอมาหนึ่งเฟรม และแปลงภาพให้เป็น base64 เพื่อให้ง่ายต่อการเก็บลงฐานข้อมูล
- บรรทัดที่ 163 จะทำการสร้างตัวแปร video โดยเป็นค่า path วิดีโอที่จะแปลงในรูปแบบของ ffmpeg_streaming
- บรรทัดที่ 164 ทำการบันทึกข้อมูลต่าง ๆ ของวิดีโอลงไปยังฐานข้อมูล
- บรรทัดที่ 166 ทำการกำหนดให้ video เป็นประเภท hls แบบ h264 ลงไปยังตัวแปร hls
- บรรทัดที่ 167 ทำการเข้ารหัสข้อมูลวิดีโอเพื่อเพิ่มความปลอดภัยของข้อมูล
- บรรทัดที่ 168 จะเป็นการแปลงวิดีโอแบบอัตโนมัติซึ่งจะแปลงวิดีโอตามค่าความละเอียดของวิดีโอ
- บรรทัดที่ 170 จะเป็นการกำหนดโฟลเดอร์ที่จะเก็บข้อมูลวิดีโอที่โดนแปลงเป็น hls แล้ว
- บรรทัดที่ 184 เมื่อการแปลงวิดีโอเสร็จสิ้นจะทำการลบไฟล์ต้นฉบับทิ้ง

```
159         if (vidData.get("videoThumbnail") == "" or vidData.get("videoThumbnail") == None):
160             b64 = getThumbnail(path)
161             vidData["videoThumbnail"] = b64
162 
163         video = ffmpeg_streaming.input(path)
164         insertVidData(vidData)
165 
166         hls = video.hls(Formats.h264())
167         hls.encryption('../key/' + vidData['encode'] + '.bin', 'http://localhost:80/hls/key/' + vidData['encode'] + '.bin')
168         hls.auto_generate_representations()
169         print("convert")
170         hls.output(
171             "..\\upload\\"
172             + vidData.get("path")
173             + "\\"
174             + vidData.get("encode")
175             + "\\"
176             + vidData.get("encode")
177             + ".m3u8",
178             monitor=lambda ffmpeg, duration, time_, time_left, process: monitor(
179                 ffmpeg, duration, time_, time_left, process, vidData["encode"]
180             ),
181         )
182         print("convert success")
183         updateVidData(vidData["videoPermit"], vidData["encode"])
184         os.remove(path)
```

ภาพประกอบที่ 3.20 ภาพประกอบโค้ดส่วนการแปลงวิดีโอและเก็บข้อมูลวิดีโอ (ต่อ)

3.9.9 โค้ดส่วนการดาวน์โหลดวิดีโอ

- บรรทัดที่ 573 และ 574 ทำการรับชื่อวิดีโอและชื่อผู้ใช้เข้ามาตามลำดับเพื่อให้รู้ path ที่จะทำการดาวน์โหลด
- บรรทัดที่ 576 กำหนดที่อยู่ไฟล์ที่ต้องการแปลงจาก hls กลับมาเป็น mp4
- บรรทัดที่ 578 กำหนดที่อยู่ที่จะจัดเก็บไฟล์แปลงเสร็จแล้วก่อนส่งคืนไปยังผู้ใช้
- บรรทัดที่ 582 ทำการกำหนดค่า response ที่จะส่งค่ากลับไปยังผู้ใช้
- บรรทัดที่ 584 ทำการส่งค่า response กลับไปยังผู้ใช้

```
571 @app.route("/download", methods=["GET"])
572 def download():
573     video = request.args.get('v')
574     user = request.args.get('u')
575 
576     video_url = 'http://localhost:80/hls/upload/'+user+'/'+video+'/'+video+'.m3u8'
577 
578     output = '../output/' + user + '_' + video + '.mp4'
579 
580     subprocess.run(['ffmpeg', '-i', video_url, output, '-y'])
581 
582     response = send_file(output, as_attachment=True)
583 
584     return response
```

ภาพประกอบที่ 3.21 ภาพประกอบโค้ดส่วนการดาวน์โหลดวิดีโอ

3.9.10 โค้ดส่วนการเก็บข้อมูล session

- บรรทัดที่ 1 จะทำการ import CryptoJS เพื่อมาเข้ารหัสข้อมูล session
- บรรทัดที่ 3 จะเป็นการกำหนด AES256 key ที่ใช้สำหรับการเข้ารหัส
- บรรทัดที่ 6 จะเป็นฟังชันก์สำหรับการเพิ่มข้อมูล session ไปยัง local storage โดยรับค่า key และ data เข้ามา
- บรรทัดที่ 7 จะทำการเข้ารหัสข้อมูลโดยเอาข้อมูลที่รับมา มาเข้ารหัสด้วย AES256 key ก่อนหน้า
- บรรทัดที่ 8 ทำการบันทึกลงไปยัง local storage
- บรรทัดที่ 13 เป็นฟังชันก์สำหรับการนำข้อมูลออกมาใช้โดยการ key ที่ต้องการค้นหาเข้ามา
- บรรทัดที่ 14 จะทำการนำข้อมูลที่เข้ารหัสใน local storage
- บรรทัดที่ 15 จะทำการถอดรหัสข้อมูลด้วย AES256 key
- บรรทัดที่ 16 ส่งคืนค่าข้อมูลที่ไม่เข้ารหัส

```
1  import CryptoJS from "crypto-js";
2
3  const encryptKey = '4yJHJpOA5AQgORVuF3VPA3xp8O2HU0Ok5k1NjPk5oWFUhGNBh4UQzoyr/lN6qvv9';
4
5  //store data to session
6  export const setlocalData = (key, data) => {
7    const jsonData = JSON.stringify(data);
8    var encryptJson = encryptData(jsonData, encryptKey);
9    localStorage.setItem(key, encryptJson );
10 };
11
12 //get data from session
13 export const getlocalData = (key) => {
14   const jsonData = localStorage.getItem(key);
15   var decryptJson = decryptData(jsonData, encryptKey);
16   // return JSON.parse(jsonData);
17   return decryptJson
18 };
```

ภาพประกอบที่ 3.22 โค้ดส่วนการเก็บข้อมูล session